

(19) 日本国特許庁 (JP)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表2004-523035

(P2004-523035A)

(43) 公表日 平成16年7月29日 (2004. 7. 29)

(51) Int. Cl.⁷G06F 13/00
H04L 12/56

F I

G06F 13/00
H04L 12/56353A
200Z

テーマコード (参考)

5B089
5K030

審査請求 有 予備審査請求 有 (全 74 頁)

(21) 出願番号 特願2002-561695 (P2002-561695)
 (86) (22) 出願日 平成13年1月31日 (2001. 1. 31)
 (85) 翻訳文提出日 平成15年7月28日 (2003. 7. 28)
 (86) 国際出願番号 PCT/11B2001/000122
 (87) 国際公開番号 W02002/061592
 (87) 国際公開日 平成14年8月8日 (2002. 8. 8)
 (81) 指定国 AP (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), EA (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), EP (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OA (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG), AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, C R, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, S G, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW

(71) 出願人 390009531
 インターナショナル・ビジネス・マシー
 ズ・コーポレーション
 INTERNATIONAL BUSI
 N E S S M A S C H I N E S C O R P O
 R A T I O N
 アメリカ合衆国 10504 ニューヨーク
 州 アーモンク ニュー オーチャード
 ロード
 (74) 代理人 100086243
 弁理士 坂口 博
 (74) 代理人 100091568
 弁理士 市位 嘉宏
 (74) 代理人 100108501
 弁理士 上野 剛史

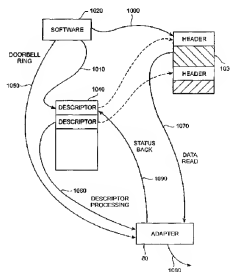
最終頁に続く

(54) 【発明の名称】 メモリを介してデータ処理システムの間でデータのフローを制御する方法および装置

(57) 【要約】

【課題】 第1および第2のデータ処理システムの間でのメモリを介するデータのフローを制御する装置を提供すること。

【解決手段】 この装置には、メモリ内の位置と第2データ処理システムの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成する記述子ロジックが含まれる。この装置には、第1および第2のデータ処理システムによるアクセスのために、記述子ロジックによって生成された記述子を保管する記述子テーブルも含まれる。



【特許請求の範囲】

【請求項 1】

メモリを介して第 1 および第 2 のデータ処理システムの間でデータのフローを制御する装置であって、メモリ内の位置と第 2 データ処理システムとの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成する記述子ロジックと、第 1 および第 2 のデータ処理システムによるアクセスのために記述子ロジックによって生成された記述子を保管する記述子テーブルとを含む装置。

【請求項 2】

記述子テーブルが、第 1 データ処理システムで保管される、請求項 1 に記載の装置。

10

【請求項 3】

記述子テーブルが、第 2 データ処理システムで保管される、請求項 1 に記載の装置。

【請求項 4】

記述子ロジックが、記述子テーブル内の別の記述子へのリンクを含む分岐記述子を生成する、請求項 1 ないし 3 のいずれかに記載の装置。

【請求項 5】

記述子テーブルが、その中の分岐記述子を介して一緒に順次リンクされる複数の記述子リストを含む、請求項 4 に記載の装置。

【請求項 6】

記述子テーブルが、循環記述子リストを含む、請求項 4 に記載の装置。

20

【請求項 7】

第 1 データ処理システムが、ホスト・コンピュータ・システムを含む、請求項 1 ないし 6 のいずれかに記載の装置。

【請求項 8】

第 2 データ処理システムが、ホスト・コンピュータ・システムとデータ通信ネットワークとの間でデータを通信するデータ通信インターフェースを含む、請求項 1 ないし 7 のいずれかに記載の装置。

【請求項 9】

メモリを有するホスト処理システムと、ホスト・コンピュータ・システムとデータ通信ネットワークとの間でデータを通信するデータ通信インターフェースと、ホスト・コンピュータ・システムのメモリとデータ通信インターフェースとの間でのデータのフローを制御する、請求項 1 ないし 8 のいずれかに記載の装置とを含むデータ処理システム。

30

【請求項 10】

メモリを介して第 1 および第 2 のデータ処理システムの間でデータのフローを制御する方法であって、記述子ロジックによって、メモリ内の位置と第 2 データ処理システムとの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成することと、第 1 および第 2 のデータ処理システムによるアクセスのために、記述子ロジックによって生成された記述子を記述子テーブルに保管することを含む方法。

【請求項 11】

記述子テーブルを第 1 データ処理システムに保管することを含む、請求項 10 に記載の方法。

40

【請求項 12】

記述子テーブルを第 2 データ処理システムに保管することを含む、請求項 10 に記載の方法。

【請求項 13】

記述子ロジックによって、記述子テーブル内の別の記述子へのリンクを含む分岐記述子を生成することを含む、請求項 10 ないし 12 のいずれかに記載の方法。

【請求項 14】

記述子テーブルを形成するために、複数の記述子リストを、分岐記述子を介して一緒に直

50

列にリンクすることを含む、請求項 13 に記載の方法。

【請求項 15】

第 1 データ処理システムが、ホスト・コンピュータ・システムを含む、請求項 10 ないし 14 のいずれかに記載の方法。

【請求項 16】

第 2 データ処理システムが、ホスト・コンピュータ・システムとデータ通信ネットワークとの間でデータを通信するデータ通信インターフェースを含む、請求項 10 ないし 15 のいずれかに記載の方法。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、ホスト・コンピュータ・システムとデータ通信ネットワークの間でデータを通信するホスト・コンピュータ・システムおよびデータ通信インターフェースなどの第 1 および第 2 のデータ処理システムの間でメモリを介してデータのフローを制御する方法および装置に関する。

【背景技術】

【0002】

従来のデータ処理ネットワークには、イーサネット（R）・アーキテクチャなどの中間のネットワーク・アーキテクチャによってすべてが相互接続される複数のホスト・コンピュータ・システムおよび複数の付加デバイスが含まれる。ネットワーク・アーキテクチャには、通常は、1 つまたは複数のデータ通信交換機が含まれる。ホスト・コンピュータ・システムおよび付加デバイスのそれぞれが、データ処理ネットワーク内のノードを形成する。各ホスト・コンピュータ・システムには、通常は、PCI バス・アーキテクチャなどのバス・アーキテクチャによって相互接続される、複数の中央処理装置およびデータ・ストレージ・メモリ・デバイスが含まれる。ネットワーク・アダプタも、ホスト・コンピュータ・システムとデータ処理ネットワーク内の他のノードの間でネットワーク・アーキテクチャを介してデータを通信するためにバス・アーキテクチャに接続される。

【発明の開示】

【発明が解決しようとする課題】

【0003】

ホスト・コンピュータ・システムとネットワーク・アーキテクチャの間のデータおよび制御情報の転送をできる限り効率的に促進することが望ましい。

【課題を解決するための手段】

【0004】

本発明によれば、メモリを介して第 1 および第 2 のデータ処理システムの間でデータのフローを制御する装置であって、メモリ内の位置と第 2 データ処理システムの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成する記述子ロジックと、第 1 および第 2 のデータ処理システムによるアクセスのために記述子ロジックによって生成された記述子を保管する記述子テーブルとを含む装置が提供される。

【0005】

記述子ロジックおよび記述子テーブルによって、ホスト・コンピュータ・システムとデータ通信ネットワークの間でデータを通信するホスト・コンピュータ・システムとデータ通信インターフェースなどの第 1 および第 2 のデータ処理システムの間でデータ・フロー制御の効率が改善される。

【0006】

記述子テーブルは、ホスト・コンピュータ・システムのメモリに保管することができる。代替案では、記述子テーブルが、データ通信インターフェースのメモリに保管される。記述子ロジックは、記述子テーブル内の別の記述子へのリンクを含む分岐記述子も生成することができる。記述子テーブルに、その中の分岐記述子を介して一緒に順次リンクされた

複数の記述子リストを含めることができる。代替案では、記述子テーブルに、循環記述子リストが含まれる。

【0007】

本発明は、メモリを有するホスト処理システムと、ホスト・コンピュータ・システムとデータ通信ネットワークの間でデータを通信するデータ通信インターフェースと、ホスト・コンピュータ・システムのメモリとデータ通信インターフェースの間でのデータのフローを制御する、上に記載の装置とを含むデータ処理システムに拡張される。

【0008】

もう1つの態様から本発明を見ると、メモリを介して第1および第2のデータ処理システムの間でデータのフローを制御する方法であって、記述子ロジックによって、メモリ内の位置と第2データ処理システムの間で通信されるデータ・パケットを定義するフレーム記述子およびメモリ内の位置を識別するポインタ記述子を含む複数の記述子を生成することと、第1および第2のデータ処理システムによるアクセスのために、記述子ロジックによって生成された記述子を記述子テーブルに保管することを含む方法が提供される。

【0009】

本発明の好ましい実施形態を、例のみとして、添付図面に関してこれから説明する。

【発明を実施するための最良の形態】

【0010】

まず図1を参照すると、本発明を実施するデータ処理ネットワークの例に、InfiniBandネットワーク・アーキテクチャ（InfiniBandはInfiniBandTrade Association社の商標である）などの中間ネットワーク・アーキテクチャ30によって相互接続された、複数のホスト・コンピュータ・システム10および複数の付加デバイス20が含まれる。ネットワーク・アーキテクチャ30には、通常は、複数のデータ通信交換機40が含まれる。ホスト・コンピュータ・システム10および付加デバイス20のそれぞれが、データ処理ネットワーク内のノードを形成する。各ホスト・コンピュータ・システム10に、PCIバス・アーキテクチャなどのバス・アーキテクチャ70によって相互接続される、複数の中央処理装置（CPU）50およびメモリ60が含まれる。ネットワーク・アダプタ80も、ホスト・コンピュータ・システム10と、データ処理ネットワーク内の他のノードとの間で、ネットワーク・アーキテクチャ30を介してデータを通信するために、バス・アーキテクチャに接続される。

【0011】

図2を参照すると、本発明の特に好ましい実施形態では、ネットワーク・アダプタ80に、ホスト・コンピュータ・システム10のバス・アーキテクチャ70への取外し可能な挿入のためのエッジ・コネクタなどのコネクタを有するプラグ可能オプション・カードが含まれる。オプション・カードは、コネクタ270を介してバス・アーキテクチャ70に接続可能な特定用途向け集積回路（ASIC）またはインテグレートッド・システム・オン・チップ（Integrated System on a Chip、ISOC）120、ISOC120に接続される1つまたは複数の第3レベル・メモリ・モジュール250、および、ネットワーク・アーキテクチャ30の媒体とISOC120の間でデータを通信するためにISOC120に接続されたインターポザ260を担持する。インターポザ260は、ネットワーク30への物理的接続を提供する。本発明のいくつかの実施形態では、インターポザ260を、単一のASIC内で実施することができる。しかし、本発明の他の実施形態では、インターポザ260を、複数の構成要素によって実施することができる。たとえば、ネットワーク30に光ネットワークが含まれる場合に、インターポザ260に、別の光トランシーバを駆動するリタイマ（retimer）を含めることができる。メモリ250は、SRAM、SDRAM、またはその組合せによって実施することができる。他の形態のメモリも、メモリ250の実施に使用することができる。ISOC120には、第1および第2のメモリが含まれる。アダプタ80のメモリ・サブシステムを、すぐに説明する。以下の説明から明らかになるように、この配置は、データ処理ネットワークで動作する分散アプリケーションの改良された性能、改良されたシステム・スケーラビリティ、ある

範囲の通信プロトコルとの互換性、およびホスト・コンピュータ・システムでの減らされた処理要件を提供する。具体的に言うと、この配置では、アダプタ 80 とホスト・システム 10 の間の異種通信プロトコルの共存が可能になる。そのようなプロトコルは、さまざまなアプリケーションをサービスし、同一のアダプタ 80 を使用し、データ構造の事前定義の組を使用することができ、これによって、ホストとアダプタ 80 の間のデータ転送が機能強化される。並列にオープンすることができ、アプリケーション・チャネルの数は、アダプタ 80 に割り振られるメモリ・リソースの量によって決定され、アダプタに組み込まれる処理能力に依存しない。下記から、複数の構成要素を単一の集積回路チップに統合するという I S O C 1 2 0 の概念によって、有利なことに、製造コストが最小になり、再利用可能なシステム基本構成要素が提供される。しかし、本発明の他の実施形態で、I S O C 1 2 0 の要素を、別個の構成要素によって実施できることも諒解されたい。

10

【0012】

以下の説明では、用語「フレーム」が、ホスト・コンピュータ・システム 10 で稼動するソフトウェアとアダプタ 80 の間で転送されるデータ単位またはメッセージを指す。各フレームに、フレーム・ヘッダおよびデータ・ペイロードが含まれる。データ・ペイロードに、ユーザ・データ、高水準プロトコル・ヘッダ・データ、肯定応答、フロー制御、またはこれらの任意の組合せを含めることができる。フレーム・ヘッダの内容を、詳細にすぐに説明する。アダプタ 80 は、フレーム・ヘッダだけを処理する。アダプタ 80 は、フレームを、ネットワーク・アーキテクチャ 30 でより効率的に移送される、より小さいパケットに断片化することができる。しかし、そのような断片化では、一般に、データ・ペイロードは変換されない。

20

【0013】

本発明の特に好ましい実施形態では、データが、ネットワーク・アーキテクチャ 30 上で、以下でパケットと称するアトミックな単位で移送される。各パケットには、経路情報と、その後のハードウェア・ヘッダ・データおよびペイロード・データが含まれる。本発明の通常の例では、1024 バイトまでのパケット・サイズが使用される。より大きいサイズのフレームは、1024 バイト・パケットに断片化される。本発明の他の実施形態で、異なるパケット・サイズを使用できることを諒解されたい。

【0014】

本発明の好ましい実施形態では、アダプタ 80 と、ホスト・コンピュータ・システム 10 で稼動する複数のアプリケーションとの間の通信が、論理通信ポート (Logical Communication Port) アーキテクチャ (LCP) を介してもたらされる。アダプタ 80 には、異なる内部データ構造へのアクセス待ち時間の最適化を可能にするメモリ階層が含まれる。このメモリ階層を、すぐに説明する。本発明の好ましい実施形態では、アダプタ 80 が、ネットワーク・アーキテクチャ 30 に宛てられたアウトバウンド (TX) データとホスト・コンピュータ・システム 10 に宛てられたインバウンド (RX) データに別々のパスを提供する。各パスには、それ自体のデータ転送エンジン、ヘッダ処理ロジック、およびネットワーク・アーキテクチャ・インターフェースが含まれる。これらのパスも、詳細にすぐに説明する。

30

【0015】

図 3 を参照すると、LCP アーキテクチャによって、ホスト・コンピュータ・システム 10 で稼動するローカル・コンシューマとアダプタ 80 の間のインターフェースのフレームワークが定義される。そのようなコンシューマの例に、アプリケーションとスレッドの両方が含まれる。コンピュータ・システム 10 を、ユーザ・アプリケーション空間 90 とカーネル空間 110 に副分割することができる。LCP アーキテクチャは、各コンシューマにネットワーク・アーキテクチャ 30 への論理ポートを与える。このポートには、ユーザ空間 90 から直接にアクセスすることができる。本発明の特に好ましい実施形態では、ハードウェア保護機構が、アクセス許可を解決する。LCP 登録が、データ・フレームの転送の前に、カーネル空間 110 によって実行される。LCP アーキテクチャでは、通信プロトコルを定義する必要がない。そうではなく、LCP アーキテクチャでは、データおよ

40

50

び制御情報を転送するための、アプリケーションとアダプタ 80 の間のインターフェースが定義される。その代わりに、通信プロトコルの詳細を、アプリケーションおよびアダプタ 80 で実行されるプログラム・コードによってセットすることができる。アダプタ 80 で使用できるチャンネルの数は、LCP 関連情報に使用可能なアダプタ・カード 80 上のメモリの量だけによって制限される。各 LCP ポートを、特定の特徴の組を有するようにプログラムすることができ。特徴の組は、特定のプロトコルに従って、ホスト・コンピュータ・システム内のメモリ 60 とアダプタ 80 の間でのデータ転送を最もよくサポートするように選択される。さまざまな通信プロトコルを、各プロトコルで異なる LCP ポートを使用して、同時にサポートすることができる。

【0016】

LCP アーキテクチャには、LCP クライアント 100、カーネル空間 110 に常駐する LCP マネージャ 130、および、アダプタ 80 に常駐する 1 つまたは複数の LCP コンテキスト 140 が含まれる。

【0017】

各 LCP クライアント 100 は、LCP ポートに接続された、単一方向アプリケーション・エンド・ポイントである。LCP クライアント 100 を、ユーザ・アプリケーション空間 90 またはカーネル 110 内に配置することができる。動作中に、各 LCP クライアント 100 は、メモリ 60 から読み取られ、アダプタ 80 によって TX LCP チャンネルを介して転送されるコマンドおよびデータを作るか、アダプタ 80 によってメモリ 60 へ RX LCP チャンネルを介して転送されるデータを消費する。

【0018】

LCP マネージャ 130 は、LCP チャンネルの割振りおよび割振り解除と、チャンネルごとのメモリ 60 内の読取／書込区域の登録の要求をサービスする、信頼される構成要素である。LCP マネージャ 130 は、他の通信動作、アプリケーション、またはホスト・コンピュータ・システム 10 のオペレーティング・システムを危険にさらさずに、ユーザ空間アプリケーションが、アダプタ 80 のリソースを使用できるようにする。

【0019】

各 LCP コンテキスト 140 は、特定の LCP クライアント 100 をサービスするのにアダプタ 80 が必要とする制御情報の組である。LCP コンテキスト 140 には、可能なコマンド、ポイント構造、およびバッファ記述子定義など、チャンネルの存在全体を通じて一定の LCP チャンネル属性を含めることができる。LCP コンテキスト 140 には、サービスを待っているデータの量、関連する LCP チャンネルにアクセスするための次のアドレスなど、LCP チャンネルに関する特定の LCP サービス情報も含まれることができる。LCP コンテキスト 140 は、アダプタ 80 に常駐するメモリに保管されて、アダプタ 80 があるチャンネルのサービスを停止し、別のチャンネルのサービスを開始する時の高速 LCP コンテキスト切替が可能になっている。

【0020】

LCP ポートの開始を要求する LCP クライアント 100 は、LCP マネージャ 130 に頼り、LCP チャンネルの割振りを要求する。LCP チャンネル属性は、この時に決定され、これによって、LCP ポートの挙動および LCP クライアント 100 が LCP ポートに関連して実行を許可される動作が規定される。LCP クライアント 100 は、一意の保護された形でアダプタ 80 にアクセスするのに使用されるアドレスを許可される。このアドレスを、ドアベル (Doorbell) アドレスと称する。

【0021】

LCP マネージャ 130 は、アダプタによる仮想アドレスから物理アドレスへの変換を可能にし、ユーザ空間クライアントが他のプログラムを改竄せずにこれらのホスト・メモリ区域にアクセスできるようにするために、ホスト・メモリ 60 の区域を登録する責任も負う。

【0022】

新しいバッファの登録および前のバッファの登録解除は、実行時中に各 LCP クライアント

10

20

30

40

50

ト 100 が要求することができる。そのような変更は、LCP クライアント 100、LCP マネージャ 130、およびアダプタ 80 の間の情報交換のシーケンスを必要とする。

【0023】

各 LCP クライアント 100 およびポートは、LCP ポートによってコマンド実行のために送られる保留中の要求をサービスするためにアダプタ 80 が必要とするすべての情報を提供する LCP コンテキスト 140 に関連する。

【0024】

LCP クライアント 100 とアダプタ 80 の間のメモリ転送を開始し、フレームの送信を開始するために、LCP クライアント 100 は、特定の動作に関する情報を保持する記述子を用意する。LCP クライアント 100 は、アダプタ 80 にマッピングされたドアベル・アドレスへの入出力書込を実行する。ドアベル・アドレスに書き込むことによって、アダプタ 80 の LCP コンテキスト 140 が更新され、新しい要求が実行のために追加される。

【0025】

アダプタ 80 は、保留中の要求を有するさまざまな送信 LCP ポートの間で調停し、次にサービスされる送信 LCP ポートを選択する。

【0026】

データの受信時に、受信されたパケットのフレームおよび LCP が、識別される。記述子を作成して、受信された LCP に必要な動作を定義する。アダプタ 80 の LCP エンジンによるこれらの記述子の実行によって、着信データが、ホスト・コンピュータ・システム 10 のメモリ 60 内で LCP チャネルに割り振られた適当なデータ・バッファに保管される。

【0027】

サービスされる LCP チャネルごとに、アダプタ 80 は、関連する LCP コンテキスト情報をロードし、この情報を使用して、データ転送の所望の組を実行する。その後、アダプタ 80 は、次に選択される LCP コンテキスト 140 の処理に継続する。

【0028】

図 4 を参照すると、前に述べたように、ISOC 120 に、第 1 メモリ空間 220 および 230 と第 2 メモリ空間 240 が含まれ、アダプタ 80 に、さらに、第 3 レベル・メモリ 250 が含まれる。第 1、第 2、および第 3 のメモリは、アダプタ 80 のメモリ・サブシステム 210 のために間隔を空ける。本発明の好ましい実施形態では、ISOC 120 に、データ送信動作専用の TX プロセッサ (TX MPC) 150 と、データ受信動作専用の RX プロセッサ (RX MPC) 160 が含まれる。本発明の特に好ましい実施形態では、プロセッサ 150 および 160 が、IBM PowerPC 405 RISC マイクロプロセッサなどの縮小命令セット・コンピューティング (RISC) マイクロプロセッサによって実施される。メモリ・サブシステム 210 内で、ISOC 120 に、第 1 および第 2 のメモリ空間の他に、TX プロセッサ 150 に関連するデータ・キャッシュ 180 および命令キャッシュ 170 が、RX プロセッサ 160 に関連する第 2 データ・キャッシュ 190 および第 2 命令キャッシュ 200 と共に含まれる。3 つのレベルの間の相違は、メモリのサイズおよび関連するアクセス時間である。すぐに明らかになるように、メモリ・サブシステム 210 は、TX プロセッサ 150 および RX プロセッサ 160 の両方による命令およびデータへの便利なアクセスと、スケーラビリティと、削減された製造コストのための TX プロセッサ 150 および RX プロセッサ 160 の間のリソースの共用とを容易にする。

【0029】

第 1 レベル・メモリ空間 (M1) 220 および 230 に、TX-M1 メモリ空間 220 および RX-M1 メモリ空間 230 が含まれる。TX-M1 メモリ 220 は、TX プロセッサ 150 によってのみアクセスでき、RX-M1 メモリ 230 は、RX プロセッサ 160 によってのみアクセスできる。動作中に、第 1 レベル・メモリ空間 220 および 230 は、一時データ構造、ハッド・テンプレート、スタックなどを保持するのに使用される。第

10

20

30

40

50

1 レベル・メモリ空間 220 および 230 の両方が、0 待ち状態に反応する。第 1 レベル・メモリ空間 220 および 230 のそれぞれは、プロセッサ 150 および 160 の対応する 1 つのデータ・インターフェースだけに接続され、命令インターフェースには接続されない。この配置によって、キャッシュ可能およびキャッシュ不能の両方の第 1 レベル・メモリ区域を使用可能にすることができるのと同時に、第 1 レベル・メモリ空間 220 および 230 内のデータへの効率的なアクセスを維持できるようになる。

【0030】

第 2 レベル・メモリ空間 (M2) 240 は、プロセッサ 150 および 160 の両方、アダプタ 80 の他の構成要素、およびホスト・コンピュータ・システム 10 から使用可能な共用メモリである。第 2 レベル・メモリ空間 240 へのアクセスは、第 1 レベル・メモリ区域 220 および 230 へのアクセスより低速である。というのは、第 2 レベル・メモリ空間 240 が、共用される内部バスを介してより多くのエージェントによって使用されるからである。第 3 レベル・メモリ空間 250 も、共用リソースである。本発明の特に好ましい実施形態では、アダプタ 80 に、その上で第 1 レベル・メモリ空間 220 および 230 と第 2 レベル・メモリ空間 240 の両方がプロセッサ 150 および 160 と同一の ASI C に集積される、コンピュータ周辺回路カードが含まれる。共用メモリ空間 240 および 250 は、一般に、高速で頻繁なアクセス・サイクルを必要としないデータ・タイプに使用される。そのようなデータ・タイプには、LCP コンテキスト 140 および仮想アドレス変換テーブルが含まれる。この共用メモリ空間 240 および 250 は、プロセッサ 150 および 160 の命令インターフェースおよびデータ・インターフェースの両方からアクセス可能である。

【0031】

アダプタ 80 は、送信データ・フローと受信データ・フローを別々に処理する。送信バスおよび受信バスのプロセッサ 150 および 160 は、タスク間の切替のオーバーヘッドを防止し、あるバス内の一時的な処理負荷を他のバスから分離し、着信データ・ストリームおよび発信データ・ストリームの処理に 2 つの組込みプロセッサを使用することを促進する。図 5 を参照すると、ISOC 120 に、送信バス・ロジック 280 および受信バス・ロジック 290 と、共用ロジック 300 が含まれる。送信バス・ロジック 280 には、各 LCP チャネルの詳細をデコードし、LCP 関連コマンドを実行のために取り出す LCP

TX エンジン 310 と、アダプタ 80 へのフレームの転送を制御する TX ロジック 320 と、TX フレームおよびパケット処理の管理用の前述の TX プロセッサ 150 と、命令および一時データ構造を保持する前述の第 1 レベル TX メモリ 220 と、リンク・ロジック 330 と、フレームのデータ・パケットへの断片化の経路指定処理などのデータ・フロー処理およびパケット処理を管理する際に TX プロセッサ 150 を支援するロジックとが含まれる。TX プロセッサ 150 は、プロセッサが例外およびエラーの際にのみ割り込まれる、ボーリングのみ方式に基づいてタスクを直列に処理する。第 1 レベル TX メモリ 220 は、プロセッサ 150 によって、TX ロジック 320 との通信に使用される。受信バス・ロジック 290 には、リンク・ロジック 340 と、着信パケットのヘッダ処理およびそのようなパケットのフレームへの変換または組立の際に前述の RX プロセッサ 160 を支援するハードウェアと、RX フレームおよびパケット処理用の前述の RX プロセッサ 160 と、命令を保持する前述の第 1 レベル RX メモリ 230 と、ネットワーク・アーキテクチャ 30 からのフレームの転送を制御する RX ロジック 350 と、各 LCP チャネルの詳細をデコードし、関連する LCP データ構造内の着信データをホスト・コンピュータ・システムのメモリ 60 に保管し、LCP クライアント 100 によってアダプタ 80 による使用のために供給される時に、空フレーム・バッファへのポインタを受け入れ、登録する、LCP RX エンジン 360 とが含まれる。RX プロセッサ 160 は、RX プロセッサ 160 が例外およびエラーの際にのみ割り込まれる、ボーリングのみ方式を使用してタスクを直列に処理する。第 1 レベル RX メモリ 230 は、RX プロセッサ 160 によって、RX ロジック 350 との通信に使用される。

【0032】

10

20

30

40

50

前に述べたように、I S O C 手法では、アダプタ 8 0 および、回路ボードおよび他のサボ
 ート・モジュールなどのアダプタの他の構成要素に関連する製造コストの削減が可能にな
 る。I S O C 手法では、アダプタ 8 0 の単純さも高まり、これによって信頼性が高まる。
 I S O C 1 2 0 の要素の間の接続の数は、効果的に無制限である。したがって、複数の幅
 広い相互接続バスを実施することができる。ホスト・コンピュータ・システム 1 0 のデ
 ータ処理オーバーヘッドを減らすために、ホスト・メモリ 6 0 との間のデータ転送動作は、
 大部分が I S O C 1 2 0 によって実行される。I S O C 1 2 0 は、着信パケットおよび
 発信パケットのヘッダの処理も実行する。送信中に、I S O C 1 2 0 は、ヘッダを作り、
 ネットワーク・アーキテクチャ 3 0 に経路指定する。受信中に、アダプタ 8 0 は、システ
 ムのメモリ内でのヘッダの位置を判定するためにヘッダを処理する。レベル 1 メモリ 2 2
 0 および 2 3 0 は、0 待ち状態メモリであり、スタック、テンプレート、テーブル、およ
 び一時保管場所などのプロセッサ・データ空間を提供する。本発明の特に好ましい実施形
 態では、送信バス・ロジック 2 8 0、受信バス・ロジック 2 9 0、および共用ロジック 3
 0 0 が、コアと称する、より小さい論理要素から作られる。用語コアが使用されるのは、
 これらの要素が、それらを異なる応用例に使用できるようにする独立型の特性を有する個
 々のロジックとして設計されるからである。

【0033】

前に示したように、送信バス・ロジック 2 8 0 は、送信フレームまたは発信フレームを処
 理する責任を負う。フレーム送信は、バス・アーキテクチャ 7 0 を介して、ホスト・コン
 ピュータ・システム 1 0 の C P U 5 0 などの C P U によって開始される。I S O C 1 2 0
 には、バス・アーキテクチャ 7 0 と通信するバス・インターフェース・ロジック 3 7 0 が
 含まれる。I S O C 1 2 0 には、バス・インターフェース・ロジック 3 7 0 を I S O C 1
 2 0 のプロセッサ・ローカル・バス (P L B) 3 9 0 に接続するバス・ブリッジング・ロ
 ジック 3 8 0 も含まれる。L C P T X エンジン 3 1 0 は、コマンドおよびフレームをホ
 スト・メモリ 6 0 から取り出す。T X プロセッサ 1 5 0 は、各フレームのヘッダを処理し
 て、ネットワーク・アーキテクチャ 3 0 上でパケットとして送信するのに適するフォー
 マットにする。T X ロジック 3 2 0 は、フレーム・データを修正なしで転送する。リンク
 ・ロジック 3 3 0 は、送信される各パケットを処理して、ネットワーク・アーキテクチャ 3
 0 0 の送信用の最終的な形にする。リンク・ロジック 3 3 0 には、それぞれがネットワー
 ク・アーキテクチャ 3 0 に接続可能である 1 つまたは複数のポートを含めることができる
 。

【0034】

前に示したように、受信バス・ロジック 2 9 0 は、着信パケットを処理する責任を負う。
 まず、ネットワーク・アーキテクチャ 3 0 から受信されたパケットが、リンク・ロジック
 3 4 0 によって処理される。リンク・ロジック 3 4 0 は、ヘッダおよびペイロードのフ
 ォーマットのパケットを再作成する。パケット・フォーマットおよびホスト・メモリ 6 0 内
 での宛先を判定するために、ヘッダが、R X プロセッサ 1 6 0 によって処理される。リン
 ク・ロジック 3 4 0 には、それぞれがネットワーク・アーキテクチャ 3 0 に接続可能であ
 る 1 つまたは複数のポートを含めることができる。R X L C P エンジンには、バス・アー
 キテクチャ 3 7 0 を介してデータをホスト・メモリ 6 0 に転送する責任を負う。

【0035】

送信バス・ロジック 2 8 0 には、T X L C P エンジン 3 1 0 と T X プロセッサ 1 5 0 の
 間の H e a d e r I n 先入れ先出しメモリ (F I F O) 4 0 0 が含まれる。受信バス・ロ
 ジック 3 4 0 には、R X プロセッサ 1 6 0 と R X L C P エンジン 3 6 0 の間の H e a d e r O
 u t F I F O 4 1 0 が含まれる。追加の F I F O およびキューを、T X ロジック 3 2 0
 内および R X ロジック 3 5 0 内に設けることができる。これらの F I F O およびキューを、
 すぐに説明する。

【0036】

共用ロジック 3 0 0 には、送信バス・ロジック 2 8 0 および受信バス・ロジック 2 9 0 に
 よって共用されるすべての要素が含まれる。この要素には、前述のバス・インターフェ
 ー

10

20

30

40

50

ス・ロジック 370、バス・ブリッジング・ロジック 380、PLB 390、第2レベル・メモリ 240、およびリモート第3レベル・メモリ 250へのアクセスを提供するコントローラ 420が含まれる。バス・インターフェース・ロジック 370は、バス・アーキテクチャ 70上でマスタおよびスレーブの両方として動作する。スレーブとして、バス・インターフェース・ロジックは、CPU 50が、第2レベル・メモリ 240に、コントローラ 420を介して第3レベル・メモリ 250に、および、ISOC 120の構成レジスタおよび状況レジスタにもアクセスできるようにする。そのようなレジスタは、一般に、CPU 50、TXプロセッサ 150、およびRXプロセッサ 160によってアクセスすることができる。マスタとして、バス・インターフェース・ロジックは、TX LCPエンジン 310およびRX LCPエンジン 360が、ホスト・コンピュータ・システム 10のメモリ 60にアクセスできるようにする。図5では、「M」がマスタ接続、「S」がスレーブ接続を示す。

【0037】

図6を参照すると、ISOC 120を通るパケットフローは、一般に対称である。言い換えると、フローの全般的な構造は、送信方向と受信方向の両方で類似する。ISOC 120は、第1インターフェース・ロジック 440と、第1制御ロジック 460と、プロセッサ・ロジック 480と、第2制御ロジック 470と、第2インターフェース・ロジック 450とを含むとみなすことができる。パケットは、下記の形で処理される。

【0038】

A. 送信方向では、情報が、バス・アーキテクチャ 70から第1インターフェース・ロジックを介してISOC 120に持ってこられる。受信方向では、情報が、ネットワーク・アーキテクチャ 30から第2インターフェース・ロジック 450を介してISOC 120に持ってこられる。

【0039】

B. 送信方向では、第1インターフェース・ロジック 440を介してISOC 120に持ってこられた情報が、第1制御ロジック 460によって処理される。受信方向では、第2インターフェース・ロジック 450を介してISOCに持ってこられた情報が、第2制御ロジック 470によって処理される。

【0040】

C. 送信方向では、フレーム・ヘッダが、第1制御ロジック 460で発信フレームについて抽出され、プロセッサ・ロジック 480によって処理される。プロセッサ・ロジック 480は、フレーム・ヘッダに基づいて、第2制御ロジック 470用の命令を生成する。発信フレームのペイロードは、第2制御ロジック 470に渡される。受信方向では、フレーム・ヘッダが、第2制御ロジック 470で着信フレームから抽出され、プロセッサ・ロジック 480によって処理される。プロセッサ・ロジック 480は、フレーム・ヘッダに基づいて、第1制御ロジック 460用の命令を生成する。着信フレームのペイロードは、第1制御ロジック 460に渡される。両方の方向で、プロセッサ 480は、ペイロード・データを直接には処理しない。

【0041】

D. 送信方向では、第2制御ロジック 470が、プロセッサ・ロジック 480から受け取った命令に従って発信ペイロード・データをパッケージする。受信方向では、第1制御ロジック 460が、プロセッサ・ロジック 480から受け取った命令に従って着信ペイロード・データをパッケージする。

【0042】

E. 送信方向では、情報が、第2インターフェース・ロジック 450からその宛先ネットワーク・アーキテクチャ 30を介して移動される。受信方向では、情報が、第1インターフェース・ロジックからその宛先バス・アーキテクチャ 70を介して移動される。

【0043】

ホスト・コンピュータ・システム 10で動作するソフトウェアへのインターフェースを、430に示す。同様に、プロセッサの入力および出力で動作するマイクロコードへのイン

ターフェースを、490および500に示す。

【0044】

図7を参照して、以下は、ISOC120を通る送信データ・フレームのフローの1例のより詳細な説明である。ISOC120は、ISOC120内の情報のさまざまなフォーマットに基づいて、LCPコンテキスト・ドメイン510、フレーム・ドメイン520、およびネットワーク・ドメイン530に分割することができる。TX LCPエンジン310には、LCP要求FIFO550、直接メモリ・アクセス(DMA)ロジック560、フレーム・ロジック580、および前述のLCPコンテキスト・ロジック140が含まれる。LCP要求FIFO550、DMAロジック560、およびLCP TXコンテキスト・ロジック590は、LCPコンテキスト・ドメイン510に常駐する。フレーム・ロジック580は、フレーム・ドメイン520に常駐する。TXロジック320、第1レベルTXメモリ空間220、およびTXプロセッサ150は、フレーム・ドメイン520とネットワーク・ドメイン530の境界にまたがる。TXリンク・ロジック330は、ネットワーク・ドメイン530に常駐する。本発明の特に好ましい実施形態では、Header in FIFO400が、第1レベルTXメモリ空間220に一体化される。一般に、ホスト・コンピュータ・システム10で実行されるアプリケーションが、フレームを作成する。フレームは、その後、アダプタ80のTX LCPチャネルを使用して送信される。アプリケーションとアダプタ80の間のハンドシェイクは、LCPマネージャ130によって前に実行された初期化を前提とする。LCPサービス要求を追加するために、LCPクライアント100が、アダプタ80に、1つまたは複数の追加の送信フレームが実行の準備ができていることを知らせる。これは、ドアップルに制御ワードを書き込むことによって実行される。ドアップルのアドレスは、LCPポートに一意に関連し、他のプロセスによるアクセスから保護されるアドレスを使用して、書込動作が、バス・アーキテクチャ70上での物理書込サイクルに変換される形で割り振られる。アダプタ80は、書込動作を検出し、特定のLCPクライアント100の前の要求の項目を増分することによって、新しい要求をログに記録する。これは、関係するLCPコンテキスト140の一部である。アダプタ80のメモリ・サブシステム210内に保持される調停リストも、更新される。単純な例では、調停で、保留中の要求を有するすべての送信LCPチャネルの間で前述のFIFO方式550が使用される。あるLCPチャネルがサービスされている間に、次のLCPチャネルが選択される。サービス・サイクルは、対応するLCPコンテキストがTX LCPエンジン310にロードされる時に開始される。その後、LCPコンテキスト140にアクセスして、LCPチャネルをサービスするアトミック・オペレーションを導出し、そのような動作のパラメータを判定する。たとえば、そのようなアトミック・オペレーションは、LCPコンテキスト140に記録されたLCPチャネル属性に基づくものとして行うことができる。完全なサービス・サイクルに、通常は、LCPクライアント100によって作成される複数のアトミックな記述子をフェッチし、実行するためにアダプタ80によって実行されるアクティビティの組が含まれる。TX LCPチャネルの場合に、サービス・サイクルに、一般に、ホスト・メモリ60からアダプタ80のメモリ・サブシステム210に複数のフレームを読み取ることが含まれる。最後に、修正を必要とするすべてのLCPコンテキスト情報(言い換えると、LCPサービス情報)が、アダプタ80のメモリ・サブシステム210内で更新される。一般に、アダプタ80によってLCPサービス・サイクル内で最初に行われる動作は、次に処理される記述子を取り出すことである。

【0045】

ISOC120による送信フレームの処理には、通常は下記のステップが含まれる。

【0046】

A. 後続LCPポート・フレーム記述子の取出

次に取り出される記述子のアドレスは、LCPチャネルのコンテキスト140の一部として保管される。アダプタ80は、ホスト・メモリ60から記述子を読み取り、LCPチャネル属性に基づいてその記述子をデコードする。記述子によって、新しいフレーム・ヘッ

10

20

30

40

50

データのサイズ、データ・ペイロードのサイズ、およびこれらの項目の位置が定義される。

【0047】

B. 仮想アドレスの物理アドレスへの変換

データ・バッファが、アプリケーション内で仮想メモリ・アドレスによって参照される場合に、そのアドレスは、アドレス変換という追加処理を受けなければならない。この場合に、アプリケーションによって使用される仮想アドレスが、アダプタ80がホスト・メモリ60にアクセスする間にアダプタ80によって使用可能な物理アドレスに変換される。これは、ページ境界を超えることを監視し、LCPマネージャ130によってアダプタ80のメモリ・サブシステム210に書き込まれる物理ページ位置情報を使用することによって行われる。仮想アドレスから物理アドレスへの変換処理は、記述子テーブルが、信頼されないLCPクライアント100によって作成される場合のセキュリティ手段としても働く。これによって、ホスト・メモリ60の無関係な区域への許可されないアクセスが防止される。

【0048】

C. フレーム・ヘッダの読取

物理アドレスリングを使用して、TXフレームのヘッダおよびペイロード・データが、ホスト・メモリ60内のバッファから読み取られる。その後、ヘッダが、TX Header In FIFO400に保管される。ヘッダ取出が完了した時に、アダプタ80は、ヘッダの処理をTXプロセッサ150によって開始できることを示す内部フラグをセットする。

【0049】

D. フレーム・データの読取

ペイロード・データが、ホスト・メモリ60から読み取られ、アダプタ80によってデータFIFO570に保管される。データFIFO570は、図7では、TXロジック320に常駐するものとして図示されている。しかし、データFIFO570を、第1レベルTXメモリ空間220に一体化することもできる。データ読取トランザクションは、送信されるデータのすべてがアダプタ80のメモリ・サブシステム210に保管されるまで継続する。読取動作の完了に続いて、状況指示が、LCPクライアント100に返される。ヘッダがHeader In FIFO400に読み取られてすぐに、ヘッダの処理を開始できることに留意されたい。データ全体が読み取られるのを待つ必要はない。

【0050】

E. フレーム・ヘッダの処理

ヘッダ処理は、TXプロセッサ150によって実行される。ヘッダ処理は、プロトコル依存であり、LCPアーキテクチャの外部のプロトコル情報が用いられる。TXプロセッサ150は、TXプロトコル・ヘッダ・マイクロコードを実行し、プロトコルおよび経路指定初期化シーケンス中にアダプタ80のメモリ・サブシステム210に既に保管された経路指定テーブルおよび他の関連情報にアクセスする。TXプロセッサ150は、新しいヘッダがHeader In FIFO400内で待っていることの指示を受け取る時に、ヘッダ処理を開始する。ヘッダ処理では、ネットワーク・アーキテクチャ30を介してパケットを送信するのに使用されるフォーマットであり、経路指定情報を含む、1つまたは複数のパケット・ヘッダが作られる。ペイロード・サイズが、ネットワーク・アーキテクチャ30によって許容される最大パケット・サイズより大きい場合には、元のペイロード・データの連続するデータ・セグメントに関してそれぞれが使用される複数のパケット・ヘッダを生成して、ネットワーク・アーキテクチャ30を介する通信用のパケットを形成することによって、ペイロードを断片化する。

【0051】

F. 送信用のパケット・ヘッダのキューイング

パケットのヘッダ・ワード数およびデータ・ワード数を定義するコマンドおよびパケット・ヘッダ自体が、TXプロセッサ150によって、第1レベル・メモリ空間220内のTX Header Out FIFO540に書き込まれる。

10

20

30

40

50

【0052】

G、送信用のパケット・ヘッダおよびパケット・データのマージ
 ネットワーク・アーキテクチャ30でのパケットの送信は、コマンドがHeader Out FIFO540内で準備ができており、データFIFO570に、関連するパケットの送信を完了するのに十分なデータが含まれる時に、必ずトリガされる。巡回冗長検査(CRC)を、各パケットのヘッダおよびデータに追加することができる。各完全なパケットが、TXリンク・ロジック330を介してネットワーク・アーキテクチャ30に転送される。

【0053】

各フレームの送信処理は、すべてのフレーム・データが、1つまたは複数のパケットによってネットワーク・アーキテクチャ30で送信された時に完了する。アダプタ80によって処理されるフレームごとに、第2のLCPクライアント100を介してアプリケーションに状況を返すことができる。この状況は、ホスト・メモリ60からアダプタ80へのフレーム・データ転送の完了、フレーム送信自体の完了、または他のレベルの送信状況とすることができる。

【0054】

どの瞬間でも、アダプタ80は、次にサービスされるLCPの選択、LCPチャネルAのサービスの開始、LCPチャネルBの最後のフレームのデータのDMA取出の実行、LCPチャネルCのフレーム・ヘッダの処理および断片化、LCPチャネルDから発するパケットの送信のうちの一部またはすべてを平行に実行することができる。

【0055】

図8を参照して、以下は、例のみとして、RX LCPポートを使用するアプリケーションによるデータ・フレーム受信の説明である。ISOC120の動作は、LCPによってサポートされるプロトコルのタイプに応じて変更することができる。アプリケーションとアダプタ80の間のハンドシェイクは、LCPマネージャ130によって前に実行された初期化を前提とする。RX LCPエンジン360には、LCP割振りロジック620、LCPコンテキスト・ロジック610、およびDMAロジック630が含まれ、これらのすべてが、LCPコンテキスト・ドメイン510に常駐する。RXプロセッサ160、第1レベルRXメモリ空間230、およびRXロジック350のすべてが、フレーム・ドメイン520とネットワーク・ドメイン530の間の境界にまたがる。RXリンク・ロジック340およびパケット援助ロジック600は、ネットワーク・ドメイン530に常駐する。本発明の特に好ましい実施形態では、Header Out FIFO410が、第1レベルRXメモリ空間230内に配置される。ISOC120によってネットワーク・アーキテクチャ30から受信されたフレームは、ホスト・メモリ60内のLCPクライアント・バッファに書き込まれる。メモリ・バッファの可用性は、LCP RXクライアント100によって判定され、着信データ・フレームの挿入のためにアダプタ80に示される。LCPクライアント100は、送信される準備ができた新しいフレームについて送信バス・ロジック280が知らされる前に述べた形に似て、ISOC120の受信ダブルに書き込むことによってバッファを提供する。ダブル・レジスタ・アドレスは、書込動作がバス・アーキテクチャ70への物理書込サイクルに変換されるように割り振られる。アダプタ80は、書込動作を検出し、特定のLCP RXクライアント100の使用可能なワード項目の数を増分することによって、空のメモリ区域の新たな提供をログに記録する。使用可能なワード・カウントは、関係するLCPコンテキスト140の一部である。アプリケーションは、バッファ内の受信したフレームの処理を完了する時に、必ずダブルに書き込む。この書込サイクルによって、新たに使用可能なメモリ空間のワード数が示される。LCPコンテキスト内のカウントが、その量だけ増分される。ネットワーク・アーキテクチャ30から受信されたパケットは、アダプタ80によってホスト・メモリ60内の連続する空間に組み立てられる、より大きいフレームの一部である場合がある。ISOC120による受信されたフレームの処理には、一般に、下記のステップが含まれる。

【0056】

10

20

30

40

50

A. パケット・ヘッダとデータの分離

RXリンク・ロジック340は、ネットワーク・アーキテクチャ30からの情報をパケットのストリームに変換する。受信されたパケットのそれぞれが、RXリンク・ロジック340によって処理されて、パケット・ヘッダがペイロード・データから分離される。ヘッダは、第1レベルRXメモリ空間230内のRX HeaderIn FIFO640にプッシュされる。ペイロードは、RXロジック350内のRXデータFIFO650にプッシュされる。RXデータFIFO650は、第1レベルRXメモリ空間230内で実施することもできる。

【0057】

B. パケット・ヘッダのデコードおよびLCPフレーム・ヘッダの生成。

パケット・ヘッダをデコードして、パケットが属するフレームのID、ペイロードのサイズ、およびフレーム・データのサイズを示すフィールドを提供する。パケット・ヘッダが、RX HeaderIn FIFO640に関するリーダになるならば、指示が、RXプロセッサ160に送られる。RXプロセッサは、パケット・ヘッダ情報を処理し、パケット・データの転送に必要な情報を含むLCP関連コマンドを生成する。そのような情報には、パケットのアドレスおよび長さが含まれる。ヘッダ処理の終りに、記述子または記述子の組が、LCP RX HeaderOut FIFO410に書き込まれ、指示がトリガされる。

【0058】

C. RX LCPコンテキスト内のデータの転送

記述子が、RX LCPエンジン360によってRX HeaderOut FIFO410から取り出され、デコードされる。記述子には、LCP番号、パケット・アドレス、パケット・データ長、および、アダプタ80のメモリ・サブシステム210に転送されるデータのソース・アドレスが含まれる。RX LCPエンジン340は、LCPコンテキスト情報を使用して、ホスト・メモリ60内の書き込まれるターゲット物理アドレス（または、ページにまたがる場合には複数のアドレス）を作成し、DMA転送を開始して、データを書き込む。

【0059】

D. ISOC DMAトランザクション

ISOC120は、適当なバス・コマンドを選択し、可能な最長のバーストを実行することによって、バス・アーキテクチャ70でのトランザクションの最適化を目指す。

【0060】

どの瞬間でも、アダプタ80は、LCPチャネルXのバッファ割振りの処理、LCPチャネルAのインバウンド・データ書込サービスの開始、LCPチャネルBのデータのDMAストアの実行、LCPチャネルC宛のパケットのフレーム組立の処理、およびLCPチャネルDのパケットの受信の一部またはすべてを並列に実行することができる。

【0061】

RXプロセッサ160およびTXプロセッサ150でのフレーム処理オーバーヘッドを最小にするために、パケット援助ロジック600に、フレーム断片化ロジック、CRCおよびチェックサム計算ロジック、およびマルチキャスト処理ロジックが含まれる。

【0062】

TX LCPエンジン310およびRX LCPエンジン360の両方とホスト10の間でデータ・フローを、これから詳細に説明する。TX LCPポートおよびRX LCPポートの両方で、データ転送用のメモリ・バッファと、そのようなメモリ・バッファをポイントする記述子構造が使用される。記述子構造は、データ・プロバイダとデータ・コンシューマの間でデータ・バッファを管理し、データ・プロバイダによって使用される空のメモリ・バッファを返すのに使用される。記述子は、物理アドレスまたは仮想アドレスのいずれかに基づいてメモリ・バッファをポイントする。

【0063】

TX LCPチャネルは、ホスト・メモリ60からISOC120のバッファへのデータ

10

20

30

40

50

転送の責任を負う。ロジックの他の層は、I S O C 1 2 0 のバッファからネットワーク 3 0 にデータを転送する責任を負う。R X L C P チャネルは、ネットワーク 3 0 から受信されたデータのホスト・メモリ 6 0 への転送の責任を負う。

【0064】

T X L C P エンジン 3 1 0 および R X L C P エンジン 3 6 0 は、比較的多数の L C P チャネルを扱うことができる。各 L C P チャネルは、それに固有のすべての情報を含むパラメータの組を有する。この情報には、チャネルの構成、現在の状態、および状況が含まれる。あるチャネルに関連する L C P コンテキスト 1 4 0 が、チャネルの初期化中に L C P マネージャ 1 3 0 によってセットされる。チャネル動作中に、L C P コンテキスト 1 4 0 の内容が、I S O C 1 2 0 のみによって更新される。L C P コンテキスト 1 4 0 は、アダプタ 8 0 のメモリ・サブシステム 2 1 0 内のコンテキスト・テーブルに保管される。L C P チャネルの L C P コンテキスト 1 4 0 へのアクセスは、L C P 番号に従って実行される。L C P の R X チャネルおよび T X チャネルでは、異なる L C P コンテキスト構造が使用される。

【0065】

データ・バッファは、ホスト 1 0 のメモリ 6 0 内のピン止めされた区域である。送信バッファには、送信用のデータが保持される。T X L C P エンジン 3 1 0 は、これらのバッファに置かれたデータを I S O C 1 2 0 の内部バッファに移動する。ネットワーク・アーキテクチャ 3 0 から受信される着信データは、R X L C P エンジン 3 6 0 によって、ホスト 1 0 のメモリ 6 0 内のバッファに移動される。バッファの所有権は、ホスト 1 0 内のソフトウェアと I S O C 1 2 0 の間で交番する。L C P T X チャネルでのイベントの順序は、次の通りである。

- A. ホスト 1 0 のソフトウェアが、ホスト 1 0 のメモリ 6 0 内で、送信されるデータと共にバッファを準備する。
- B. ソフトウェアが、バッファ内のデータが送信の準備ができていることを I S O C 1 2 0 に通知する。
- C. I S O C 1 2 0 が、バッファからデータを読み取る。
- D. I S O C 1 2 0 が、ホスト 1 0 のソフトウェアに対して、読み取られ、新しいデータを転送するためにホスト 1 0 のソフトウェアによって再利用することができるバッファを識別する。

【0066】

L C P R X チャネルでのイベントの順序は、次の通りである。

- A. ホスト 1 0 のソフトウェアが、I S O C 1 2 0 が受信されたデータを書き込むことができるバッファを準備する。
- B. ソフトウェアが、空きバッファがホストのメモリ 6 0 内で準備ができていることを I S O C 1 2 0 に通知する。
- C. I S O C 1 2 0 が、データをバッファに書き込む。
- D. I S O C 1 2 0 が、ホスト 1 0 のソフトウェアに対して、受信されたデータが書き込まれ、ソフトウェアによって処理されることができるバッファを識別する。

【0067】

ソフトウェアが、I S O C 1 2 0 によって使用されるバッファを準備する時に、バッファ情報が、ダブル・レジスタを介して追跡される。I S O C 1 2 0 によって使用されるバッファに関する情報は、状況更新を使用してまたは完了キューを介して、ソフトウェアに返される。T X L C P チャネルについて、バッファに、T X L C P エンジン 3 1 0 によって I S O C 1 2 0 に転送され、処理されて、ネットワーク 3 0 での送信用の 1 つまたは複数のパケットになるデータおよびヘッダの情報が含まれる。ヘッダは、I S O C 1 2 0 の T X プロセッサ 1 5 0 によって、ネットワーク 3 0 で送信されるパケットのヘッダを生成するのに使用される。R X L C P チャネルについて、空きバッファが、ホスト 1 0 のソフトウェアによってアダプタ 8 0 に割り当てられる。アダプタ 8 0 は、受信されたパケットをバッファに書き込む。

10

20

30

40

50

【0068】

記述子によって、I S O C 1 2 0 およびホスト 1 0 のソフトウェアの両方に既知のデータ構造が定義されている。ソフトウェアは、記述子を使用して、I S O C 1 2 0 に制御情報を転送する。制御情報は、所望の機能に応じて、フレーム記述子、ポインタ記述子、または分岐記述子の形とすることができる。ソフトウェア内および I S O C 1 2 0 内の記述子ロジックは、行われる制御手段に従って記述子を生成し、修正する。そのような手段を、すぐに説明する。フレーム記述子には、パケットの記述子（たとえば、データ長、ヘッダ長など）が含まれる。ポインタ記述子には、データ位置の記述子が含まれる。分岐記述子には、記述子位置の記述（たとえば、記述子のリンク・リスト）が含まれる。記述子内の情報は、T X L C P エンジン 3 1 0 および R X L C P エンジン 3 6 0 によって実行されるデータ移動動作の、ホスト 1 0 内のソフトウェアによる制御に使用される。T X パケット・ヘッダを生成するためにフレームを処理するのに使用される情報は、フレームのヘッダに配置される。図 9 を参照すると、記述子を、単一のテーブル 7 0 0 内に設けることができ、L C P コンテキスト 1 4 0 が、テーブル 7 0 0 の先頭をポインタする。図 1 0 を参照すると、記述子を、リンクされた記述子テーブル 7 2 0 から 7 4 0 の構造で配置することもできる。L C P チャネル初期化の後に、L C P コンテキスト 1 4 0 は、構造内の最初の記述子テーブル 7 2 0 の先頭をポインタする。分岐記述子 7 5 0 から 7 7 0 は、テーブル 7 2 0 から 7 4 0 のリンク・リストを生成するのに使用され、ここで、記述子テーブル 7 2 0 から 7 4 0 の終りの分岐記述子 7 5 0 から 7 7 0 は、別のテーブル 7 2 0 から 7 4 0 の始めをポインタする。図 9 に戻ると、分岐記述子は、循環バッファを生成するのにも使用することができ、ここで、テーブル 7 0 0 の終りの分岐記述子 7 1 0 は、同一のテーブル 7 0 0 の始めをポインタする。循環バッファは、受信バスで使用することもできる。この場合に、L C P コンテキスト 1 4 0 が、バッファの先頭をポインタするように開始される。バッファは、I S O C 1 2 0 がその終りに達した時に、ラップ・アラウンドされる。ホスト 1 0 のソフトウェアは、ホスト 1 0 のメモリ 6 0 に（受信バスと送信バスの両方について）、またはアダプタ 8 0 のメモリ 2 5 0 に（送信バスのみについて）記述子を書き込むことができる。アダプタ 8 0 のメモリ・サブシステム 2 1 0 への記述子の書込には、ホスト 1 0 のソフトウェアによる入出力動作が含まれ、アダプタ 8 0 のメモリ・サブシステム 2 1 0 が占有される。ホスト 1 0 のメモリ 6 0 での記述子の書込は、アダプタ 8 0 が、新しい記述子を読み取らなければならない時に、必ずホスト 1 0 のメモリ 6 0 にアクセスすることを必要とする。ソフトウェア記述子の位置は、L C P チャネルごとに独立に、L C P マネージャ 1 3 0 によって定義される。記述子の位置は、システム性能最適化に従って定義される。記述子は、キューの構成での柔軟性を提供する。

【0069】

T X L C P エンジン 3 1 0 および R X L C P エンジン 3 6 0 は、記述子テーブル内の記述子にアクセスし、データ・バッファにアクセスするのに、アドレスを使用する。アドレスは、物理アドレスまたは仮想アドレスのいずれかとすることができる。用語物理アドレスは、I S O C 1 2 0 が、そのままバス 7 0 に駆動できるアドレスを指す。用語仮想アドレスは、物理アドレスではなく、ソフトウェアまたはマイクロコードによって使用されるアドレスを指す。仮想アドレスは、物理アドレスを生成するために、マッピングを受けなければならない。T X L C P エンジン 3 1 0 および R X L C P エンジン 3 6 0 によって使用されるアドレスは、L C P チャネル・コンテキスト 1 4 0 内のポインタ、ホスト 1 0 で稼動するソフトウェアによって準備される記述子内のポインタ、R X プロセッサ 1 6 0 によって準備される記述子内のポインタ、および T X プロセッサ 1 5 0 によって準備される記述子内のポインタ（完了メッセージを返すのに使用される）という異なるソースを有することができる。ポインタは、記述子またはデータ・バッファをポインタすることができる。T X L C P エンジン 3 1 0 および R X L C P エンジン 3 6 0 によって使用されるすべてのアドレスを、任意選択として、バス 7 0 上の物理アドレスとして使用される新しいアドレスにマッピングすることができる。アドレス・マッピングは、T X L C P エンジン 3 1 0 および R X L C P エンジン 3 6 0 によって行われる。I S O C 1 2

10

20

30

40

50

0 は、ローカル・メモリ 210 を使用して、変換テーブルを保持する。LCP マネージャ 130 は、メモリ登録中にアダプタ 80 に変換テーブルを書き込む。アドレス・マッピングによって、バッファまたは記述子テーブルに仮想アドレスを使用できるようになる。仮想アドレッシングによって、物理的に複数の物理ページに配置される仮想バッファの管理が可能になる。アドレス・マッピングによって、ホスト 10 が、ソフトウェアのための変換プロセッサを必要とせずに、仮想アドレスを使用するアプリケーションを直接に扱えるようになる。

【0070】

図 11 を参照すると、この図には、ホスト 10 のソフトウェアから見えるバッファ 880 のイメージ 800 が示されている。ホスト 10 内のメモリ 60 にアクセスするのに使用される、アドレスの物理マッピング 810 も示されている。仮想ポインタが、バッファ内の位置を 820 ポイントする。この例のバッファは、ホスト 10 のメモリ 60 内の少数の連続しないページ 840 から 870 を占める仮想バッファである。LCP エンジン 310 および 360 は、変換テーブル 830 を介してアドレスを変換することによってマッピングを実行する。変換テーブルは、仮想バッファ 880 からマッピングされる各物理バッファ 840 から 870 の先頭への物理アドレス・ポインタを保持する。アダプタ 80 内でのアドレス・マッピングによって、ホスト 10 のメモリ 60 内で記述子およびデータ・バッファをマッピングする時の柔軟性が可能になる。アダプタ 80 内でのアドレス・マッピングによって、ホスト 10 のソフトウェアが物理アドレスへのアドレス変換を実行することを必要とせずに、仮想アドレスを使用するソフトウェア・バッファへの直接接続も可能になる。

【0071】

アダプタ 80 がホストのメモリ 60 に書き込む各バケットは、それに関連する状況を有する。状況を用いると、アダプタ 80 とホスト 10 のソフトウェアの間の同期化が可能になる。状況は、バケットの異なる信頼性レベルを示すのに使用することができる。ISOC 120 は、下記の状況ライト・バックを提供する：送信 DMA 完了は、TX バケット内のデータがアダプタ 80 に読み込まれたことを示し；信頼される送信は、ネットワーク 30 でのデータ送信の完了を示すために返され；受信 DMA 完了は、メモリ 60 への受信データ転送の完了を示し；信頼される受信は、ネットワーク 30 内の宛先ノードによる送信バケットの受信を示す。

【0072】

TX フレーム記述子には、2 バイトの状況フィールドが含まれる。状況ライト・バックは、トランザクション状態が記述子にライト・バックされることを意味する。状況には、ホスト 10 のソフトウェアがボーリングできる完了ビットが含まれる。ホスト 10 のソフトウェアは、セットされた完了ビットを見つけた時に、そのフレーム記述子によって定義されるフレームに関連するバッファを再利用することができる。

【0073】

完了キューは、RX LCP チャネルによって実施される。完了キューによって使用される LCP チャネルは、どの RX LCP チャネルによっても実施することができる柔軟性および特性のすべてを有する。TX プロセッサ 150 および RX プロセッサ 160 は、状況ライト・バックを生成して、信頼される送信、信頼される受信、受信 DMA 完了、または送信 DMA 完了を示す。フレームに関連する異なる指示を、異なる場合に使用することができる。たとえば、信頼される送信の場合に、TX プロセッサ 150 は、バケット送信の状況を示す内部レジスタを読み取る。信頼される受信の場合に、RX プロセッサ 160 が、肯定応答を含む受信されたバケットとして完了指示を得る。受信 DMA 完了の場合に、RX プロセッサ 160 が、フレーム完了情報を使用する。送信 DMA 完了の場合に、TX プロセッサ 150 が、アダプタ 80 での送信用フレームの受信を示す。完了キューは、単一の TX LCP チャネルまたは単一の RX LCP チャネルによって使用することができる、あるいは、複数のチャネルによって共用することができる。アダプタ 80 のマイクロ・コードが、RX LCP エンジン 360 のコマンド・キューへのフレーム記述子を開

10

20

30

40

50

始することによって、状況キューを更新する。図12を参照すると、状況は、完了キュー920を含む完了状況LCP900を介してホスト10のメモリ60に転送される。完了キュー920は、連続し（物理的にまたは仮想的にのいずれか）、ホスト10のメモリ60内に配置される。たとえば、完了キューを、連続するバッファ内に保持することができる。完了キューの項目930は、固定されたサイズを有することが好ましい。各項目が、受信LCP910に関連するバッファ950の先頭へのポインタ940を保持する。バッファ950には、完了状況に関連するパケット960が書き込まれる。

【0074】

TXソフトウェア／アダプタ・ハンドシェイクに、TX LCPポートおよび完了RX LCPポートが含まれる。各LCP送信チャネルでは、下記のデータ構造が使用される。メモリ・マップされたアドレスとして実施され、記述子およびデータを処理する増分要求についてアダプタ80に知らせる、ドアベル項目。各プロセスは、ドアベル・アクセスに使用されるメモリ・マッピングされたアドレスの単一のページへの一意のアクセスを有する。

LCP属性フィールドおよび状況フィールドを含む、アダプタ・メモリ空間210内のLCPコンテキスト項目。

送信記述子の構造。この構造は、ホスト10のメモリ60内の複数の物理ページにまたがることができる。仮想アドレッシングが、記述子について使用される場合に、変換テーブルを使用して、あるページから次のページに移動する。物理アドレッシングが、記述子について使用される場合に、分岐記述子が、あるページから次のページへの移動に使用される。送信記述子には、アダプタ80への記述子関連データのすべての転送の後に更新することができる状況フィールドが含まれる。

ポインタ記述子によってポイントされる、ホスト10のメモリ60内でピン止めされた送信データ・バッファ。仮想アドレッシングが、データ・バッファについて使用される場合に、変換テーブルによって、ポインタが、アダプタ80によってホスト10のメモリ60にアクセスするのに使用される物理アドレスに変換される。アダプタ・メモリ空間210内の変換テーブルおよび保護ブロックが、アドレス・マッピングに使用される。

【0075】

図13を参照すると、送信パケットのフローに、ステップ1000で、ホスト10のソフトウェア1020が、送信されるデータをバッファ1030に書き込むことが含まれる。ステップ1010で、ソフトウェア1020が、記述子1040を更新する。記述子1040は、ホスト10のメモリ60内またはアダプタ80のメモリ・サブシステム210内のいずれかとすることができる。ステップ1050で、ソフトウェア1020が、ドアベルを鳴らして、新しいデータの送信の準備ができていることをアダプタ80に通知する。ステップ1060で、アダプタ80が、異なるLCPチャネルからの要求の間の調停を管理する。あるチャネルが調停に勝った時に、アダプタ80が、新しい記述子1040を読み取る。ステップ1070で、アダプタ80が、データを読み取る。ステップ1080で、データが、ネットワーク30に送信される。ステップ1090で、状況が、記述子1040内または完了キュー内で更新される。

【0076】

TX LCPチャネルでは、データ・バッファにアクセスする時にアドレス変換を使用することができる。この場合に、データ・バッファは、複数のメモリ・ページからなる。この処理に関する限り、これらのメモリ・ページは、連続する仮想メモリ空間内にある。しかし、アダプタ80に関する限り、これらのメモリ・ページは、連続しない物理メモリ空間内にある場合がある。完了状況構造に、送信されたフレームの状況を示す情報が含まれる。これは、別々のLCPチャネルとして実施される。すべてのフレームの最初の記述子であるフレーム記述子は、フレームがアダプタ80に転送された後に更新することができる任意選択の状況フィールドを有する。

【0077】

10

20

30

40

50

図14を参照すると、送信LCPチャネル・フローの例で、記述子1100が、ホスト10のメモリ60に配置される。記述子1100およびパケット1120を保管するバッファ1110へのアクセスは、アダプタ80に配置される変換テーブル1130を介するアドレス変換を必要とする。バッファ1110によって、ホスト10のソフトウェアの仮想アドレス空間内の連続するスペースが使用される。各フレーム1120は、2タイプの記述子すなわち、パケットに関係する情報を与えるフレーム記述子1140と、データ1120を保持するバッファ1110をポイントするポインタ記述子1150とによって記述される。各パケットに、データ・ペイロード1170と、同一のバッファ1180でそれに先行するヘッダ1160が含まれる。

【0078】

ドアップルへの書込トランザクション1190によって、アダプタ80による使用に使用可能なワード数1200が更新される。この情報は、LCPコンテキスト140に保管される。送信LCPコンテキスト140には、送信されるデータを保持するバッファ1110の先頭へのポインタ1210が含まれる。LCPチャネルが、ISOC120の内部調停に勝つ時に、ISOC120は、LCPコンテキスト140内のポインタ1210に従ってLCPチャネルの記述子を読み取る。LCPチャネルの記述子1100およびバッファ1110の両方に関連する仮想アドレスは、アダプタ80のメモリ・サブシステム210内に配置される変換テーブル1130を使用して物理アドレスに変換される。変換テーブル1130は、メモリ・バッファの登録中にLCPマネージャ130によって更新される。ISOC120は、データおよびフレーム・ヘッダをバッファ1110からアダプタ80へ読み取る。フレーム・ヘッダ1160が、ISOC上でネットワーク30のヘッダに置換される1320。パケット・ヘッダおよび対応するデータが、ネットワーク30に送信される。

【0079】

RX LCPポートは、着信データをISOC120からホスト10で稼動するソフトウェア・アプリケーションによって使用されるメモリ60に転送するのに使用される。TX LCPチャネルは、完全に、ホスト10のソフトウェアによって開始される記述子を介して制御される。RX LCPチャネルでは、ホスト10のソフトウェアおよびISOC120の両方からの記述子が使用される。ISOC120によって開始される記述子は、LCPチャネル動作を制御して、ホスト10のメモリ60内の受信されたフレームの宛先を定義するのに使用される。ホスト10のソフトウェアによって開始される記述子は、バッファが変換テーブル内のマッピングを介して定義されない場合に、バッファの位置を定義するのに使用することができる。ホスト10のソフトウェアとアダプタ80の間のハンドシェイクを実施するために、2つのLCPチャネルすなわち、受信された着信データ構造を扱うRX LCPチャネルと、完了状況キューを扱うRX LCPチャネルが使用されることが好ましい。この完了状況は、アダプタ80が、ホスト10のソフトウェアに、ホスト10のメモリ60へのフレーム転送が完了したことをシグナリングするのに使用される。項目が、完了キュー構造の順次アドレスに挿入される。完了状況項目のそれぞれに、アダプタ80によってマークされ、項目所有権がアダプタ80からホスト10のソフトウェアに転送されたことを検査するためにホスト10のソフトウェアによってボーリングされるフィールドが含まれる。1つまたは複数のRX LCPチャネルが、同一の完了状況キューを使用することができる。複数のRX LCPチャネルによる完了状況キューの共用は、ISOC120によって実行される。

【0080】

RX LCPチャネルは、着信パケットの宛先アドレスを示す情報を必要とする。ISOC120は、空きバッファの位置を見つけるための下記の2つのアドレッシングを有する。

直接アドレッシング・モードは、バッファをポイントするのにポインタ記述子を使用しないLCPチャネルを指す。宛先アドレスは、ISOC120のマикроコードによって定義されるか、コンテキスト140から読み取られる。

10

20

30

40

50

間接アドレッシング・モードは、記述子構造内でデータ・バッファへのポインタを維持する LCP チャネルを指す。記述子は、ホスト 10 のメモリ 60 内に配置されることが好ましい。

【0081】

直接アドレッシングでは、アダプタ 80 を介する着信パケットの処理の待ち時間が、実質的に短縮される。しかし、直接アドレッシングでは、アダプタ 80 での仮想アドレスから物理アドレスへの変換情報の保管を含めて、LCP マネージャ 130 によるメモリ・バッファの登録が必要である。ホスト 10 のソフトウェアは、チャネル・ドアベルに書き込んで、チャネルによって使用可能な空きバッファに追加されるワード数を示す。直接モードでは、下記のステップを使用して、宛先バッファのアドレスを判定する。

- A. アドレス A が、LCP エンジンへのコマンドとして駆動される。
- B. (任意選択) アドレス A をアドレス A' にマッピングする。
- C. アドレス A' (ステップ B が実行される場合) または A (ステップ B が実行されない場合) が、宛先バッファの基底アドレスになる。

【0082】

間接モードでは、アダプタ 80 が、記述子を使用して、データ・バッファのアドレスを見つける。記述子は、ホスト 10 のソフトウェアによって管理される。記述子は、ホスト 10 のメモリ 60 に配置されることが好ましい。用語「間接」は、アダプタ 80 が、宛先アドレスを定義するために追加情報を読み取ることを強調するのに使用される。アダプタ 80 は、実行時中にこの情報にアクセスする。間接アドレッシングでは、変換テーブルを保管するのに必要なアダプタ 80 のメモリの量が削減される。記述子は、通常は、ホスト 10 のメモリ 60 に配置される。間接モードでは、宛先バッファのアドレスを判定するのに、下記のステップが使用される。

- A. アドレス A が、LCP エンジンに対するコマンドとして駆動される。
- B. (任意選択) アドレス A をアドレス A' にマッピングする。
- C. アドレス A' (ステップ B が実行される場合) または A (ステップ B が実行されない場合) が、ポインタ記述子のアドレスになる。
- D. バッファへのポインタすなわちアドレス B を、記述子から読み取る。
- E. (任意選択) アドレス B をアドレス B' にマッピングする。
- F. アドレス B' (ステップ E が実行される場合) または B (ステップ E が実行されない場合) が、宛先バッファの基底アドレスになる。

【0083】

各 R X LCP チャネルでは、下記のデータ構造が使用される。メモリ・マッピングされたアドレスとして実施される、ドアベルへのアクセスによって、追加データについてまたはアダプタ 80 がパケット・データを書き込むのに使用可能な記述子について、アダプタ 80 が知らされる。

アダプタ 80 のメモリ・サブシステム 210 内の LCP コンテキスト項目に、LCP 属性、状況、構成、および状況フィールドが含まれる。

間接モードで使用されるメモリ・バッファをポイントする記述子。

ホスト 10 のメモリ 60 内の連続する仮想アドレス空間内のバッファ。

アドレス・マッピング用の、アダプタ 80 のメモリ空間 210 内の変換テーブルおよび保護ブロック。

【0084】

パケットの受信のフローは、下記の特性に依存する。

直接または間接の、アドレッシング・モード。

間接モードについて、記述子がホスト 10 のメモリ 60 内に配置される。

直接モードについて、アドレス・マッピングを、記述子へのアクセス中に使用してもしなくてもよい。

アドレス・マッピングを、バッファへのアクセス中に使用してもしなくてもよい。

間接モードについて、アドレス保護を、記述子へのアクセス中に使用してもしなくてもよい

い。

アドレス保護を、バッファへのアクセス中に使用してもしなくてもよい。

【0085】

これらの特性は、LCPチャネルごとに、LCPチャネル初期化中にチャネルのコンテキスト140の一部としてセットされる。

【0086】

図15を参照すると、パケット受信のフローに、ステップ1300での、受信されたデータ用の空きバッファ1320の、ホスト10内のソフトウェア1310による準備が含まれる。ステップ1330で、間接モードで、ホスト10のソフトウェア1310が、記述子1340を更新する。記述子1340は、ホスト10のメモリ60に配置される。ステップ1350で、ホスト10のソフトウェアが、ドアベルを鳴らして、空きバッファ空間についてアダプタ80に通知する。間接モードでは、ドアベルが、新しい記述子1340を示す情報を提供する。直接モードでは、ドアベルが、追加された空きバッファ空間を示す情報を提供する。この段階で、アダプタ80は、ネットワーク30からホスト10のメモリ60に受信データを転送する準備ができている。ステップ1300、1330、および1350は、ホスト10のソフトウェア1310が、RX LCPチャネルに空きバッファ1320を追加する時に、必ず繰り返される。ISOC120は、受信されたパケットごとに下記のステップを繰り返す。ステップ1360で、アダプタ80が、データを受信する。ステップ1370で、間接モードで、アダプタ80が、空きデータ・バッファ1320の位置をポイントする記述子1340を読み取る。ステップ1380で、データおよびヘッダが、データ・バッファ1320に書き込まれる。ステップ1390で、状況が、完了キュー内で更新される。

【0087】

図16を参照すると、受信LCPチャネル・フローの例で、ポインタ記述子が使用されない。さらに、変換テーブルが使用されない。データ・バッファ1400で、バッファ1400を使用するホスト10内のソフトウェアの物理アドレス空間の連続する空間が使用される。ヘッダおよびデータ・ペイロードの両方が、バッファ1400に書き込まれる。ドアベルへの書込トランザクション1410によって、アダプタ80による使用のために使用可能なデータ空間が更新される。この情報は、LCPコンテキスト140に保管される。受信/完了LCPコンテキスト140に、新しいデータ/完了項目の書込に使用される次/現在のアドレスへの、バッファ1400の先頭へのポインタ1420とオフセット1430が含まれる。アダプタ80は、パケットを受信する時に、次のパケット位置へのオフセット1430を増分し、使用可能なデータ空間を更新する。完了項目1440が、フレーム受信の完了時、フレーム・タイムアウト時、またはLCPクライアント100が知ることを必要とする他のフレーム・イベントについて、完了LCP1450に追加される。完了項目1440には、LCPデータ・バッファ1400内でフレームを突き止めるためにLCPクライアント100が必要とするすべての情報が含まれる。ホスト10のソフトウェアは、完了項目1440内のフィールドを使用して、完了項目1440の所有権を与えられていることを認識する。

【0088】

ISOC120は、アダプタ80のメモリ・サブシステム210とホスト10のメモリ60の間でデータを移動するのにLCPチャネルを使用できるようにする。ホスト10のメモリ60からアダプタ80にデータを転送するために、送信チャネルが使用される。アダプタ80からホスト10のメモリ60にデータを転送するために、受信チャネルが使用される。データが、ホスト10のメモリ60からアダプタ80に転送される時に、フレーム記述子に、ISOC120のバス390での宛先アドレスが含まれる。このアドレスによって、フレーム・データ・ペイロードの宛先が定義される。パケット・ヘッダは、通常形で転送される。これによって、ISOC120のメモリ空間へのテーブルおよびコードのロードが可能になる。ISOC120のメモリ空間からホスト10のメモリ60に受信チャネルを介してデータを転送するために、記述子が、RXプロセッサ160によって開

10

20

30

40

50

始される。この記述子には、ホスト 10 のメモリ 60 での宛先アドレスとソース・アドレスの両方を示す情報が含まれる。

【0089】

上で説明した本発明の好ましい実施形態では、アダプタ 80 が、バス・アーキテクチャ 70 を介してホスト・コンピュータ・システム 10 の CPU 50 およびメモリ 60 に接続される。しかし、本発明の他の実施形態では、アダプタ 80 を、バス・アーキテクチャ 70 とは独立にホスト・コンピュータ・システム 10 に統合することができる。たとえば、本発明の他の実施形態で、アダプタ 80 を、ホスト・メモリ 60 に接続されたメモリ・コントローラを介してホスト・システムに統合することができる。

【0090】

さらに、上で説明した本発明の好ましい実施形態では、アダプタ 80 が、ホスト・コンピュータ・システム 10 に挿入されるプラグ可能アダプタ・カードの形で実施された。しかし、本発明の他の実施形態で、アダプタ 80 の異なる実施形態が可能であることを認識されたい。たとえば、アダプタ 80 を、CPU 50 およびメモリ 60 と共に、ホスト・コンピュータ・システムのマザー・ボード上に配置することができる。

【図面の簡単な説明】

【0091】

【図 1】データ処理ネットワークの例のブロック図である。

【図 2】データ処理ネットワークのネットワーク・インターフェース・アダプタ・カードのブロック図である。

【図 3】データ・ネットワークのホスト・システムの例のブロック図である。

【図 4】ネットワーク・アダプタ・カードのインテグレートッド・システム・オン・ア・チップ (ISOC) の例のブロック図である。

【図 5】ISOC のもう 1 つのブロック図である。

【図 6】ISOC を通る情報のフローを示す ISOC のブロック図である。

【図 7】ISOC を通る論理送信バスのブロック図である。

【図 8】ISOC を通る論理受信バスのブロック図である。

【図 9】循環記述子テーブルのブロック図である。

【図 10】記述子テーブルのリンクされた組のブロック図である。

【図 11】仮想バッファおよびその物理的に対応するバッファのブロック図である。

【図 12】完了キューのブロック図である。

【図 13】ホストからネットワークへのデータの送信フローのブロック図である。

【図 14】ホストからネットワークへのデータの送信フローのもう 1 つのブロック図である。

【図 15】ネットワークからホストへのデータの受信フローのブロック図である。

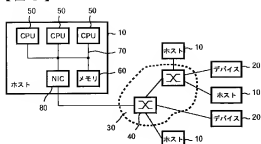
【図 16】ネットワークからホストへのデータの受信フローのもう 1 つのブロック図である。

10

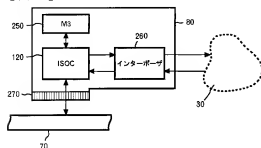
20

30

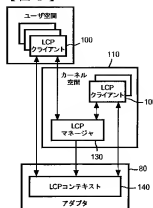
【図 1】



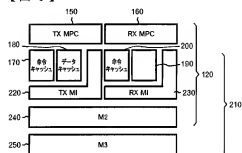
【図 2】



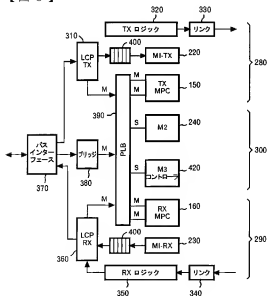
【図 3】



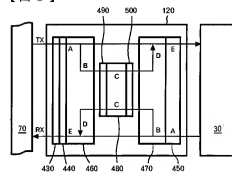
【図 4】



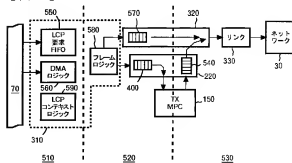
【図 5】



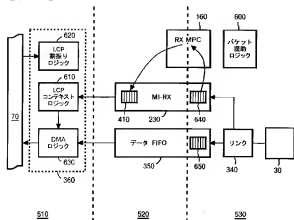
【図 6】



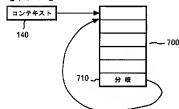
【図 7】



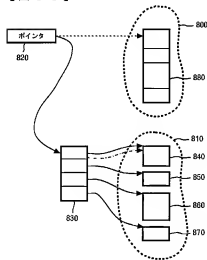
【図 8】



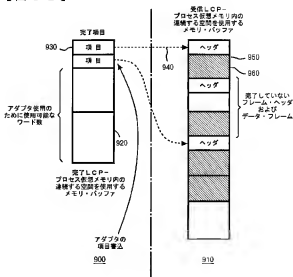
【図 9】



【図 11】



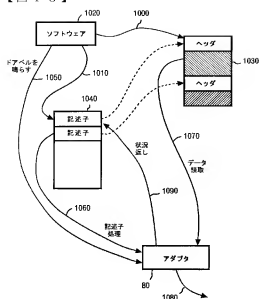
【図 12】



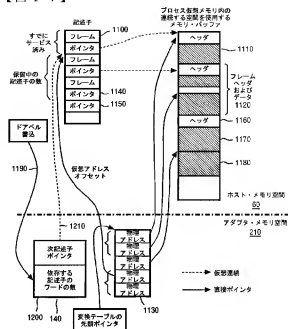
【図 10】



【図 13】



【図 14】



METHOD AND APPARATUS FOR CONTROLLING FLOW OF DATA BETWEEN
DATA PROCESSING SYSTEMS VIA A MEMORY

The present invention relates to a method and apparatus for controlling flow of data via a
5 memory between first and second data processing systems such as a host computer system and
a data communications interface for communicating data between the host computer system
and a data communications network.

A conventional data processing network comprises a plurality of host computer systems and a
10 plurality of attached devices all interconnected by an intervening network architecture such as
an Ethernet architecture. The network architecture typically comprises one or more data
communications switches. The host computer systems and the attached devices each form a
node in the data processing network. Each host computer system typically comprises a
15 plurality of central processing units and data storage memory device interconnected by a bus
architecture such as a PCI bus architecture. A network adapter is also connected to the bus
architecture for communicating data between the host computer system and other nodes in the
data processing network via the network architecture. It would be desirable for transfer of
data and control information between the host computer system and the network architecture
to be facilitated as efficiently as possible.

20 In accordance with the present invention, there is now provided apparatus for controlling flow
of data between first and second data processing systems via a memory, the apparatus
comprising: descriptor logic for generating a plurality of descriptors including a first
descriptor defining a data packet to be communicated between a location in the memory and
25 the second data processing system, and a pointer descriptor identifying the location in the
memory; and a descriptor table for storing the descriptors generated by the descriptor logic for
access by the first and second data processing systems.

The descriptor logic and descriptor table improve efficiency of data flow control between the
30 first and second data processing systems such as a host computer system and a data
communications interface for communicating data between the host computer system and a
data communications network.

WO 02/061592

- 2 -

PCT/IB00/00122

The descriptor table may be stored in the memory of the host computer system. Alternatively, the descriptor table is stored in a memory of the data communications interface. The descriptor logic may also generate a branch descriptor comprising a link to another descriptor in the descriptor table. The descriptor table may comprise a plurality of descriptor lists sequentially linked together via branch descriptors therein. Alternatively, the descriptor table may comprise a cyclic descriptor list.

The present invention extends to a data processing system comprising a host processing system having a memory, a data communications interface for communicating data between the host computer system and a data communications network, and apparatus as hereinbefore described for controlling flow of data between the memory of the host computer system and the data communications interface.

Viewing the present invention from another aspect, there is now provided a method for controlling flow of data between first and second data processing systems via a memory, the method comprising: by descriptor logic, generating a plurality of descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and storing the descriptors generated by the descriptor logic in a descriptor table for access by the first and second data processing systems.

Preferred embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of an example of a data processing network;

Figure 2 is a block diagram of a network interface adapter card for the data processing network;

Figure 3 is a block diagram of an example of a host computer system for the data network;

WO 02/061592

- 3 -

PCT/IB99/00122

Figure 4 is a block diagram of an example of an Integrated System on a Chip (ISOC) for the network adapter card;

Figure 5 is another block diagram of the ISOC;

5

Figure 6 is a block diagram of the ISOC demonstrating information flow through the ISOC;

Figure 7 is a block diagram of a logical transmit path through the ISOC;

10

Figure 8 is a block diagram of a logical receive path through the ISOC;

Figure 9A is a block diagram of a cyclic descriptor table;

15

Figure 9B is a block diagram of a linked set of descriptor tables;

Figure 10 is a block diagram of a virtual buffer and its physical counterpart buffer;

Figure 11 is a block diagram of a completion queue;

20

Figure 12 is a block diagram of a transmit flow of data from the host to the network;

Figure 13 is another block diagram of a transmit flow of data from the host to the network;

Figure 14 is a block diagram of a receive flow of data from the network to the host; and,

25

Figure 15 is another block diagram of a receive flow of data from the network to the host.

Referring first to Figure 1, an example of a data processing network embodying the present invention comprises a plurality of host computer systems 10 and a plurality of attached

30

devices 20 interconnected by an intervening network architecture 30 such as an InfiniBand network architecture (InfiniBand is a trade mark of the InfiniBand Trade Association). The network architecture 30 typically comprises a plurality of data communications switches 40,

WO 02/061592

- 4 -

PCT/JP00/06122

The host computer systems 10 and the attached devices 20 each form a node in the data processing network. Each host computer system 10 comprises a plurality of central processing units (CPUs) 50, and a memory 60 interconnected by a bus architecture 70 such as a PCI bus architecture. A network adapter 80 is also connected to the bus architecture for communicating data between the host computer system 10 and other nodes in the data processing network via the network architecture 30.

Referring now to Figure 2, in particularly preferred embodiments of the present invention, the network adapter 80 comprises a plugable option card having a connector such as an edge connector for removable insertion into the bus architecture 70 of the host computer system 10. The option card carries an Application Specific Integrated Circuit (ASIC) or Integrated System on a Chip (ISOC) 120 connectable to the bus architecture 70 via the connector 170, one or more third level memory modules 250 connected to the ISOC 120, and an interposer 260 connected to the ISOC 120 for communicating data between the nodes of the network architecture 30 and the ISOC 120. The interposer 260 provides a physical connection to the network 30. In some embodiments of the present invention, the interposer 260 may be implemented as a single ASIC. However, in other embodiments of the present invention, the interposer 260 may be implemented by multiple components. For example, if the network 30 comprises an optical network, the interposer 260 may comprise a retimer driving a separate optical transceiver. The memory 250 may be implemented by SRAM, SDRAM, or a combination thereof. Other forms of memory may also be employed in the implementation of memory 250. The ISOC 120 includes a first and a second memory. The memory subsystem of the adapter 80 will be described shortly. As will become apparent from the following description, this arrangement provides: improved performance of distributed applications operating on the data processing network; improved system scalability; compatibility with a range of communication protocols and reduced processing requirements in the host computer system. More specifically, this arrangement permits coexistence of heterogeneous communication protocols between the adapters 80 and the host systems 10. Such protocols can serve various applications, use the same adapter 80, and use a predefined set of data structures thereby enhancing data transfers between the host and the adapter 80. The number of application channels that can be opened in parallel is determined by the amount of memory resources allocated to the adapter 80 and is independent of processing power embedded in the

WO 02/061592

- 5 -

PCT/JP00/06122

adapter. It will be appreciated from the following that the ISOC 120 concept of integrating multiple components into a single integrated circuit chip component advantageously minimizes manufacturing costs and provides reusable system building blocks. However, it will also be appreciated that in other embodiments of the present invention, the elements of the ISOC 120 may be implemented by discrete components.

In the following description, the term "frame" refers to data units or messages transferred between software running on the host computer system 10 and the adapter 80. Each Frame comprises a Frame Header and a data payload. The data payload may contain user data, high level protocol header data, acknowledgments, flow control or any combination thereof. The contents of the Frame Header will be described in detail shortly. The adapter 80 processes only the Frame Header. The adapter 80 may fragment Frames into smaller packets which are more efficiently transported on the network architecture 30. However, such fragmentation generally does not transform the data payload.

In particularly preferred embodiment of the present invention, data is transported on the network architecture 30 in atomic units hereinafter referred to as Packets. Each Packet comprises route information followed by hardware header data and payload data. In a typical example of the present invention, a packet size of up to 1024 bytes is employed. Frames of larger size are fragmented into 1024 byte packets. It will be appreciated that in other embodiments of the present invention, different packet sizes may be employed.

In a preferred embodiment of the present invention, communications between the adapter 80 and multiple applications running on the host computer system 10 are effected via a Logical Communication Port architecture (LCP). The adapter 80 comprises a memory hierarchy which allows optimization of access latency to different internal data structures. This memory hierarchy will be described shortly. In preferred embodiments of the present invention, the adapter 80 provides separate paths for outbound (TX) data destined for the network architecture 30 and inbound (RX) data destined for the host computer system 10. Each path includes its own data transfer engine, header processing logic and network architecture interface. These paths will also be described in detail shortly.

Referring now to Figure 3, the LCP architecture defines a framework for the interface between local consumers running on the host computer system 10 and the adapter 80. Examples of such consumers include both applications and threads. The computer system 10 can be subdivided into a user application space 90 and a kernel space 110. The LCP architecture provides each consumer with a logical port into the network architecture 30. This port can be accessed directly from a user space 90. In particularly preferred embodiments of the present invention, a hardware protection mechanism takes care of access permission. An LCP registration is performed by in the kernel space 110 prior to transfer of data frames. The LCP architecture need not define a communication protocol. Rather, it defines an interface between the applications and the adapter 80 for transfer of data and control information. Communication protocol details may be instead set by the application and program code executing in the adapter 80. The number of channels that can be used on the adapter 80 is limited only by the amount of memory on the adapter card 80 available for LCP related information. Each LCP port can be programmable to have a specific set of features. The set of features is selected according to the specific protocol to best support data transfer between the memory 60 in the host computer system and the adapter 80. Various communication protocols can be supported simultaneously, with each protocol using a different LCP port.

The LCP architecture comprises LCP Clients 100, an LCP Manager 130 resident in the kernel space 130, and one or more LCP Contexts 140 resident in the adapter 80.

Each LCP Client 100 is a unidirectional application and point connected to an LCP port. An LCP client 100 can be located in the user application space 90 or in the kernel 110. In operation, each LCP client 100 produces commands and data to be read from the memory 60 and transferred by the adapter 80 via a TX LCP channel, or consumes data transferred by the adapter 80 to the memory 60 via an RX LCP channel.

The LCP Manager 130 is a trusted component that services request for LCP channel allocations and deallocations and for registration of read/write access in the memory 60 for each channel. The LCP Manager 130 allows a user space application to use resources of the adapter 80 without compromising other communication operations, applications, or the operating system of the host computer system 10.

WO 02/061592

- 7 -

PCT/JP00/06122

Each LCP Context 140 is the set of control information required by the adapter 80 to service a specific LCP Client 100. The LCP Context 140 may include LCP channel attributes which are constant throughout existence of the channel, such as possible commands, pointer structure, and buffer descriptor definitions. The LCP Context 140 may also include specific LCP service information for the LCP channel, such as the amount of data waiting for service, and the next address to access for the related LCP channel. The LCP context 140 is stored in memory resident in the adapter 80 to enable fast LCP context switching when the adapter 80 stops servicing one channel and starts servicing another channel.

10 An LCP Client 100 requiring initiation of an LCP port turns to the LCP Manager 130 and requests the allocation of an LCP channel. The LCP channel attributes are determined at this time and prescribe the behavior of the LCP port and the operations that the LCP Client 100 is authorized to perform in association with the LCP port. The LCP Client 100 is granted an
15 address that will be used to access the adapter 80 in a unique and secure way. This address is known as a Doorbell Address.

The LCP Manager 130 is also responsible for registering areas of the host memory 60 to enable virtual to physical address translation by the adapter, and to allow userspace clients to
20 access these host memory areas without tampering with other programs.

Registration of new buffers and deregistration of previous buffers can be requested by such LCP Client 100 during run-time. Such a change requires a sequence of information exchange between the LCP Client 100, the LCP Manager 130, and the adapter 80.

25 Each LCP Client 100 and port are associated with an LCP Context 140 that provides all the information required by the adapter 80 to service pending requests sent by the LCP port for command execution.

30 To initiate memory transfers between the LCP Client 100 and the adapter 80, and initiate transmission of frames, the LCP Client 100 prepares descriptors holding the information for a specific operation. The LCP Client 100 then performs an I/O write to the Doorbell address

WO 02/061592

- 8 -

PCT/IB03/00122

supplied to the adapter 80. Writing to the Doorbell address updates the LCP Context 140 on the adapter 80, adding the new request for execution.

The adapter 80 arbitrates between various transmit LCP ports that have pending requests, and selects the next one to be serviced.

On receipt of data, the Frame and LCP for a received packet are identified. Descriptors are generated to define the operation required for the receive LCP. Execution of these descriptors by an LCP Engine of the adapter 80 stores the incoming data in an appropriate data buffer allocated to the LCP channel in the memory 60 of the host computer system 10.

For each LCP channel serviced, the adapter 80 loads the associated LCP context information and uses this information to perform the desired set of data transfers. The adapter 80 then continues on to process the next selected LCP Context 140.

Referring now to Figure 3, and as mentioned earlier, the ISOC 120 comprises a first memory space 220 and 230 and a second memory space 240 and the adapter 80 further comprises a third level memory 250. The first, second, and third memory spaces form part of a memory subsystem 210 of the adapter 80. In a preferred embodiment of the present invention, the ISOC 120 comprises a TX processor (TX MPC) 150 dedicated to data transmission operations and an RX processor (RX MPC) 160 dedicated to data reception operation. In particularly preferred embodiments of the present invention, processors 150 and 160 are implemented by Reduced Instruction Set Computing (RISC) microprocessors such as IBM PowerPC 405 RISC microprocessors. Within the memory subsystem 210, the ISOC 120 comprises, in addition to the first and second memory spaces, a data cache 180 and an instruction cache 170 associated with TX processor 150, together with a second data cache 190 and second instruction cache 190 associated with RX processor 160. The difference between the three levels is the size of memory and the associated access time. As will become apparent, shortly, the memory subsystem 210 facilitates convenient access to instruction and data by both the TX processor 150 and the RX processor 160; scalability; and sharing of resources between the TX processor 150 and the RX processor 160 in the interests of reducing manufacturing costs.

The first level memory spaces (M1) 220 and 230 comprise a TX-M1 memory space 220 and RX-M1 memory space 230. The TX-M1 memory 220 can be accessed only by the TX processor 150 and the RX-M1 memory 230 can be accessed only by the RX processor 160. In operation the first level memory spaces 220 and 230 are used to hold temporary data structures, header templates, stacks, etc. The first level memory spaces 220 and 230 both react to zero wait states. Each one of the first level memory spaces 220 and 230 is connected only to the data interface of the corresponding one of the processors 150 and 160 and not to the instruction interface. This arrangement enables both cacheable and non-cacheable first level memory access available while maintaining efficient access to data in the first level memory spaces 230 and 240.

The second level memory space (M2) 240 is a shared memory available to both processors 150 and 160, other components of the adapter 80, and to the host computer system 10. Access to the second level memory space 240 is slower than access to the first level memory areas 220 and 230 because the second level memory space 240 is used by more agent via a shared internal bus. The third level memory space 250 is also a shared resource. In particularly preferred embodiments of the present invention the adapter 80 comprises a computer peripheral circuit card on which the first level memory spaces 220 and 230 and the second level memory space 240 are both integrated on the same ASIC as the processors 150 and 160. The shared memory spaces 240 and 250 are generally used for data types that do not require fast and frequent access cycles. Such data types include LCP contexts 140 and virtual address translation tables. The shared memory spaces 240 and 250 are accessible to both instruction and data interfaces of the processors 150 and 160.

The adapter 80 handles transmission and reception data flows separately. The separate processor 150 and 160 for the transmission and reception path avoids the overhead of switching between tasks, isolates temporary processing loads in one path from the other path, and facilitates use of two embedded processors to process incoming and outgoing data streams. Referring now to Figure 5, the ISOC 120 comprises transmission path logic 280 and reception path logic 300, and shared logic 300. The transmission path logic 280 comprises an LCP TX engine 310 for decoding specifics of each LCP channel and fetching LCP related

WO 02/061592

- 10 -

PCT/JP00/06122

commands for execution; TX logic 320 for controlling transfer of frames into the adapter 80, the aforementioned TX processor 150 for managing TX frame and packet processing; the aforementioned first level TX memory 230 for holding instructions and temporary data structures; and link logic 330; and logic for assisting the TX processor 150 in managing the data flow and packet processing such as receiving processing for fragmentation of frames into data packets. The TX processor 150 processes tasks in series based on a polling only scheme in which the processor is interrupted only on exceptions and errors. The first level TX memory 220 is employed by the processor 150 for communicating with TX logic 320. The reception path logic 290 comprises link logic 340; hardware for assisting the aforementioned RX processor 160 in processing headers of incoming packets and transformation or assembly of such packets into frames; the aforementioned RX processor 160 for RX frame and packet processing; the aforementioned first level RX memory 230 for holding instructions; RX logic 350 for controlling transfer of frames from the network architecture 30; and an LCP RX engine 360 for decoding the specifics of each LCP channel, storing the incoming data in the related LCP data structures in the memory 60 of the host computer system, and accepting and registering pointers to empty frame buffers as they are provided by the LCP Client 100 for use by the adapter 80. The RX processor 160 processes tasks in series using a polling only scheme in which the RX processor 160 is interrupted only on exceptions or errors. The level 1 RX memory 230 is used by the RX processor 160 to communicate with the RX logic 350.

As mentioned earlier, the ISOC approach permits reduction in manufacturing costs associated with the adapter 80 and the other components thereof, such as the circuit board and the other supporting modules. The ISOC approach also increases simplicity of the adapter 80, thereby increasing reliability. The number of connections between elements of the ISOC 120 is effectively unlimited. Therefore, multiple *one wide* interconnect paths can be implemented. In the interests of reducing data processing overheads in the host computer system 10, data transfer operations to and from the host memory 60 are predominantly performed by the ISOC 120. The ISOC 120 also performs processing of the header of incoming and outgoing packets. During transmission, the ISOC 120 builds the header and routes it to the network architecture 30. During reception, the adapter 80 processes the header in order to determine its location in the system's memory. The level 1 memories 220 and 230 are zero wait state memories providing processor data space such as stack, templates, tables, and temporary storage

WO 02/061592

- 11 -

PCT/IB00/06122

locations. In especially preferred embodiments of the present invention, the transmission path logic 280, reception path logic 290, and shared logic 300 are built from multiple logic elements referred to as cores. The term core is used because these elements are designed as individual pieces of logic which have stand-alone properties enabling them to be used for different applications.

As indicated earlier, the transmission path logic 280 is responsible for processing transmission or outgoing frames. Frame transmission is initiated via the bus architecture 70 by a CPU such as CPU 50 of the host computer system 10. The ISOC 120 comprises bus interface logic 370 for communicating with the bus architecture 70. The ISOC 120 also comprises bus bridging logic 390 connecting the bus interface logic 370 to a processor local bus (PLB) 380 of the ISOC 120. The TX LCP engine 310 fetches commands and frames from the host memory 60. The TX processor 150 processes the header of each frame into a format suitable for transmission as packets on the network architecture 30. The TX logic 320 transfer the frame data without modification. The link logic 330 processes each packet to be transmitted into a final form for transmission on the network architecture 30. The link logic 330 may comprises one or more ports each connectable to the network architecture 30.

As indicated earlier, the reception path logic 290 is responsible for processing incoming packets. Initially, packets received from the network architecture 30 are processed by link logic 340. Link logic 340 reconstructs the packet in a header and payload format. To determine the packet format and its destination in the host memory 60, the header is processing by the RX processor 230. The link logic 340 may comprises one or more ports each connectable to the network architecture 30. The RX LCP engine is responsible for transferring the data into the host memory 60 via the bus architecture 70.

The transmission path logic 280 comprises a HeaderIn first-in-first-out memory (FIFO) 400 between the TX LCP engine 310 and the TX processor 220. The reception path logic comprises a HeaderOut FIFO 410 between the RX processor 230 and the RX LCP engine 360. Additional FIFOs and queues are provided in the TX logic 320 and the RX logic 330. These FIFOs and queues will be described shortly.

The shared logic 300 comprises all logical elements shared by the transmission path logic 280 and the reception path logic 290. These elements include the aforementioned bus interface logic 370, bus bridge logic 380, PLB 390, second level memory 240 and a controller 420 for providing access to the remote third level memory 250. The bus interface logic 370 operates as both master and slave on the bus architecture 70. As a slave, the bus interface logic allows the CPU 50 to access the second level memory 240, the third level memory 250 via the controller 420, and also configuration registers and status registers of the ISOC 120. Such registers can generally be accessed by the CPU 50, the TX processor 150 and the RX processor 160. As a master, the bus interface logic allows the TX LCP engine 310 and the RX LCP engine 360 to access the memory 60 of the host computer system 10. In Figure 5, "M" denotes a master connection and "S" denotes a slave connection.

Referring now to Figure 6, packet flow through the ISOC 120 is generally symmetrical. In other words, the general structure of flow is similar in both transmit and receive directions.

- 15 The ISOC 120 can be regarded as comprising first interface logic 440, a first control logic 460, processor logic 480, second control logic 470, and second interface logic 450. Packets are processed in the following manner:
- A. In the transmit direction, information is brought into the ISOC from the bus architecture 70 through the first interface logic. In the receive direction, information is brought into the ISOC 120 from the network architecture 30 through the second interface logic 450.
 - 20 B. In the transmit direction, information brought into the ISOC 120 through the first interface logic 440 is processed by the first control logic 460. In the receive direction, information brought into the ISOC through the second interface logic 450 is processed by the second control logic 470.
 - 25 C. In the transmit direction, a frame header is extracted for an outgoing frame at the first control logic 460 and processed by the processor logic 480. The processor logic 480 generates instructions for the second control logic 470 based on the frame header. The payload of the outgoing frame is passed to the second interface logic 470. In the
 - 30

WO 02/061592

- 12 -

PCT/IB95/00122

- receive direction, a frame header is extracted from an incoming frame at the second control logic 470 and processed by the processor logic 480. The processor logic 480 generates instructions for the first control logic 460 based on the frame header. The payload of the incoming frame is passed to the first control logic 460. In both directions, the processor 480 is not directly handling payload data.
- 5
- D. In the transmit direction, the second control logic 470 packages the outgoing payload data according to the instructions received from the processor logic 480. In the receive direction, the first control logic 460 packages the incoming payload according to the instructions received from the processor logic 480.
- 10
- E. In the transmit direction, the information is moved through the second interface logic 450 to its destination via the network architecture 50. In the receive direction, the information is moved through the first interface logic to its destination via the bus architecture 70.
- 15

An interface to software operating on the host computer system 10 is shown at 430. Similarly, interfaces to microcode operating on the processor inputs and outputs is shown at 490 and 500.

- 20
- Referring to Figure 7, what follows now is a more detailed description of one example of a flow of transmit data frames through the ISOC 120. The ISOC 120 can be divided into an LCP context domain 510, a frame domain 520 and a network domain 530 based on the various formats of information within the ISOC 120. The TX LCP engine 310 comprises an LCP request FIFO 550, Direct Memory Access (DMA) logic 560, frame logic 580, and the aforementioned LCP context logic 140. The LCP request FIFO 550, DMA logic 560, and LCP TX Context logic 590 reside in the LCP context domain 510. The frame logic 590 resides in the frame domain 520. The TX logic 320, first level TX memory space 220, and TX processor 150 straddle the boundary between the frame domain 520 and the network domain 530. The TX link logic 330 resides in the network domain 530. In particularly preferred embodiments of the present invention, the Host/Link FIFO 400 is integral to the first level TX memory space 220. In general, an application executing on the host computer system 10 creates a frame. The

frame is then transmitted using a TX LCP channel on the adapter 80. Handshaking between the application and the adapter 80 assumes a prior initialization performed by the LCP Manager 130. To add an LCP Service Request, an LCP Client 100 informs the adapter 80 that one or more additional transmit frames are ready to be executed. This is performed by writing to a control word in the Doorbell. The Doorbell's addresses are allocated in such a way that the write operation is translated into a physical write cycle on the bus architecture 70, using an address that is uniquely associated with the LCP port and protected from access by other processes. The adapter 80 detects the write operation and logs the new request by incrementing an entry of previous requests for the specific LCP Client 100. This is part of the related LCP Context 140. An arbitration list, retained in the memory subsystem 210 of the adapter 80 is also updated. In a simple example, arbitration uses the aforementioned FIFO scheme 550 between all transmit LCP channels having pending requests. While one LCP channel is serviced, the next LCP channel is selected. The service cycle begins when the corresponding LCP Context is loaded into the TX LCP engine 310. The LCP Context 140 is then accessed to derive atomic operations for servicing the LCP channel and to determine permissions for such operations. For example, such atomic operations may be based on LCP channel attributes recorded in the LCP Context 140. A complete service cycle typically includes a set of activities performed by the adapter 80 to fetch and execute a plurality of atomic descriptors created by the LCP Client 100. In the case of a TX LCP channel, the service cycle generally includes reading multiple frames from the host memory 60 into the memory subsystem 210 of the adapter 80. Upon conclusion, all the LCP Context information requiring modification (in other words, the LCP Service Information) is updated in the memory subsystem 210 of the adapter 80. In general, the first action performed by the adapter 80 within the LCP Service cycle, is to fetch the next descriptor to be processed.

Processing of transmission frames by the BSOC 120 typically includes the following steps:

- A. Fetching the subsequent LCP port frame descriptor:
 - The address of the next descriptor to be fetched is stored as part of the LCP channel's Context 140. The adapter 80 reads the descriptor from host memory 60 and decodes

WO 02/061592

- 15 -

PCT/IB03/00122

the descriptor based on the LCP channel attributes. The descriptor defines the size of the new frame header, the size of the data payload, and the location of these items.

B. Conversion of virtual address to physical address.

- 5 If a data buffer is referenced by virtual memory addresses in an application, the address should go through an additional process of address translation. In this case, the virtual address used by the application is translated into a physical address usable by the adapter 80 while it access the host memory 60. This is done by consulting page
- 10 book/entry crossings and using physical page location information written by the LCP manager 130 into the memory subsystem 210 of the adapter 80. The virtual to physical translation process serves also as a security measure in cases where a descriptor table is created by an LCP client 100 which is not trusted. This prevents unauthorized access to unrelated areas of the host memory 60.

15 C. Reading the frame header.

Using physical addressing, the header and payload data of the TX frame are read from buffers in the host memory 60. The header is then stored in the TX Header in FIFO 400. When the header fetch is completed, the adapter 80 sets an internal flag indicating that processing of the header can be initiated by the TX processor 150.

20

D. Reading the frame data.

- The payload data is read from the host memory 60 and stored by the adapter 80 in a data FIFO 570. The data FIFO 570 is shown in Figure 7 as resident in the TX logic 320. However, the data FIFO 570 may also be integral to the first level TX memory space 220. Data read transactions continue until all data to be transmitted is stored in the memory subsystem 210 of the adapter 80. Following completion of the read operation, a status indication is returned to the LCP Client 100. Note that processing of the header can start as soon as the header has been read into the Header FIFO 400.
- 25
- 30 There is no need to wait for the whole data to be read.

WO 02/061592

- 16 -

PCT/IB99/00122

B. Processing the frame header

The header processing is performed by the TX processor 150. Header processing is protocol dependent and involves protocol information external to the LCP architecture. The TX processor 150 runs TX protocol header microcode and accesses routing tables and other relevant information already stored in the memory subsystem 210 of the adapter 80 during a protocol and routing initialization sequence. When the TX processor 150 receives an indication that a new header is waiting in the HeaderIn FIFO 400, it starts the header processing. The header processing produces one or more packet headers which are in the format employed to send packets over the network architecture 30 and include routing information. If the payload size is larger than a maximum packet size allowed by the network architecture 30, the payload is fragmented by generating several packet headers each used in connection with consecutive data segments of the original payload data to form packets for communication over the network architecture 30.

F. Queuing the packet header for transmission

A command defining the number of header words and the number of data words for a packet and the packet header itself are written by the TX processor 150 to a TX HeaderOut FIFO 540 in the first level memory space 220.

G. Merging packet header and packet data for transmission.

Transmission of a packet on the network architecture 30 is triggered whenever a command is ready in the HeaderOut FIFO 540, and the data FIFO 570 contains enough data to complete the transmission of the related packet. A Cyclic Redundancy Check (CRC) may be added to the header and data of each packet. Each complete packet is transferred to the network architecture 30 via the TX link logic 330.

The transmission process for each frame is completed when all the frame data is transmitted on the network architecture 30, by means of one or more packets. For each frame processed by

the adapter 80, a status may be returned to the application via a second LCP Client 100. This status indicates the completion of the frame data transfer from the host memory 60 onto the adapter 80, completion of the frame transmission itself, or other levels of transmission status.

- 5 At any instance in time, the adapter 80 may be concurrently executing some or all of the following actions: selecting the next LCP to be serviced; initiating services for LCP channel A; executing DMA fetch of data for the last frame of LCP channel B; processing a frame header and fragmentation for LCP channel C; and, interleaving packets originated by LCP channel D.
- 10 Referring to Figure 8, what follows now, by way of example only, is a description of a data frame reception by an application using an RX LCP port. The operation of the ISOC 120 may vary depending on the type of protocol supported by the LCP. Handshaking between the application and the adapter 80 assumes a prior initialization performed by the LCP manager
- 15 130. The RX LCP engine 360 comprises LCP allocation logic 620, LCP Context logic 610, and DMA logic 630 all residing in the LCP domain 520. The RX processor 160, first level RX memory space 230, and RX logic 350 all straddle the boundary between the frame domain 520 and the network domain 530. The RX link logic 340 and packet assist logic 600 reside in the network domain 530. In particularly preferred embodiments of the present invention, the
- 20 HeaderOut FIFO 410 is located in the first level RX memory space 230. Frames received by the ISOC 120 from the network architecture 30 are written into LCP client buffers in the host memory 60. Availability of memory buffers is determined by the LCP RX client 100 and is indicated to the adapter 80 for insertion of incoming data frames. The LCP client 100 provides buffers by writing into a receive Doorbell on the ISOC 120, similar to the aforementioned
- 25 manner in which the transmission path logic 280 is informed of new frames ready to be transmitted. The Doorbell register address is allocated such that the write operation is translated into a physical write cycle on the bus architecture 70. The adapter 80 detects the write operation and logs the new provision of empty memory areas by incrementing the number of available word entries for the specific LCP RX Client 100. The available word
- 30 count is part of the related LCP context 140. Whenever an application completes processing of a received frame within a buffer, it writes to the Doorbell. The write cycle indicates the number of words in the newly available memory space. The count within the LCP context is

WO 02/061592

- 18 -

PCT/JP00/00122

incremented by that amount. A packet received from the network architecture 30 may be part of a larger frame that will be assembled by the adapter 80 into contiguous space in the host memory 60. Processing of received frames by the ISOC 100 generally includes the following steps:

5

A. Splitting packet header and data

The RX link logic 540 transmits information from the network architecture 30 into a stream of packets. Each received packet is processed by the RX link logic 540 to separate the packet header from the payload data. The header is pushed into an RX HeaderIn FIFO 640 in the first level RX memory space 230. The payload is pushed into an RX data FIFO 650 in the RX logic 350. The RX data FIFO 650 may also be implemented in the first level RX memory space 230.

15 B. Decoding the packet header and generating and LCP frame header.

The packet header is decoded to provide fields indicative of an ID for the frame to which the packet belongs, the size of the payload, and the size of the frame data. Once the packet header is read for the RX HeaderIn FIFO 640, an indication is sent to the RX processor 150. The RX processor processes the packet header information and generates an LCP related command including information required to transfer the packet data. Such information includes packet address and length. At the end of the header processing, a descriptor, or a set of descriptors, are written to the LCP RX HeaderOut FIFO 410, and an indication is triggered.

25

C. Transfer of data within the RX LCP Context.

The descriptors are fetched from the RX HeaderOut FIFO 410 by the RX LCP engine 360, and then decoded. The descriptors include the LCP number, packet address, packet data length and the source address of the data to be transferred in the memory subsystem 240 of the adapter 80. The RX LCP engine 340 uses the LCP Context.

30

WO 02/061592

- 19 -

PCT/IB95/00122

information to create a target physical address (or addresses if a page is crossed) to be written to in the host memory 60 and initiates DMA transfers to write the data.

D. ISOC DMA transactions.

5

The ISOC 120 aims to optimize transactions on the bus architecture 70 by selecting appropriate bus commands and performing longest possible bursts.

At any instance in time, the adapter 80 may be concurrently executing some or all of the following: processing a buffer allocation for LCP channel X; initiating an inbound data write service for LCP channel A; executing a DMA store of data for LCP channel B; processing a frame assembly of a packet destined for LCP channel C; and, receiving packets for LCP channel D.

15 To minimize frame processing overhead on the RX processor 160 and TX processor 150, packet assist logic 600 comprises frame fragmentation logic, CRC and checksum calculation logic, and multicast processing logic.

The data flow between both the TX and RX LCP engines 310 and 360 and the host 10 will now be described in detail. Both TX and RX LCP ports use memory buffers for transferring data and descriptor structures that point to such memory buffers. The descriptor structures are used to administer data buffers between a data provider and a data consumer and to return empty memory buffers to be used by the data provider. The descriptors point to the memory buffers based on either physical or virtual addresses.

25

TX LCP channels are responsible for data transfer from the host memory 60 into buffers of the ISOC 120. Other layers of logic are responsible for transferring data from buffers of the ISOC 120 into the network 30. RX LCP channels are responsible for transferring data received from the network 30 to the host memory 60.

30

The TX and RX LCP engines 310 and 360 are capable of handling a relatively large number of LCP channels. Each LCP channel has a set of parameters containing all information

WO 02/061592

- 20 -

PCT/JP00/06122

separate secrets. The information comprises the configuration of the channel, current state and status. The LCP context 140 associated with a channel is set by the LCP manager 130 during initialization of the channel. During channel operation, the content of the LCP context 140 is updated only by the ISOC 120. The LCP contexts 140 are saved in a context table within the memory subsystem 210 of the adapter 80. Access to the LCP context 140 of an LCP channel is performed according to the LCP number. The LCP-RX and TX channels use different LCP context structures.

Data buffers are pinned areas in the memory 60 of the host 10. Transmit buffers hold data that for transmission. The TX LCP engine 310 moves the data located in these buffers into internal buffers of the ISOC 120. Incoming data received from the network 30 is moved by the RX LCP engine 360 into buffers in the memory 60 of the host 10. Ownership of the buffers alternates between software in the host 10 and the ISOC 120. The order of events on LCP TX channels is as follows:

- 15 A. Software in the host 10 prepares buffers with data to be transmitted in the memory 60 of the host 10;
- B. The software notifies the ISOC 120 that data in the buffers is ready to be transmitted;
- C. The ISOC 120 reads the data from the buffers; and,
- 20 D. The ISOC 120 identifies to the software in the host 10 the buffers that were read and can be reused by the software in the host 10 to transfer new data.

The order of events on LCP RX channels is as follows:

- 25 A. The software in the host 10 prepares buffers into which the ISOC 120 can write the received data;
- B. The software notifies the ISOC 120 that free buffers are ready in the memory 60 of the host;
- C. The ISOC 120 writes the data to the buffers; and,
- 30 D. The ISOC 120 identifies to the software in the host 10 the buffers that were filled with received data and can be processed by the software.

WO 02/061592

- 21 -

PCT/JP00/06122

- When the software prepares buffers to be used by the ISOC 120, buffer information is tracked via doorbell registers. Information relating to buffers used by the ISOC 120 is returned to the software using a status update or through a completion queue. For TX LCP channels, the buffers include data and header information transferred by the TX LCP engine 310 into the ISOC 120 and processed to become one or more packets for transmission on the network 30.
- The header is used by the TX processor 150 of the ISOC 120 to generate the header of the packet to be transmitted on the network 30. For RX LCP channels, free buffers are assigned by the software in the host 10 to the adapter 80. The adapter 80 fills the buffers with the received packets.
- The descriptors have defined data structures known to both the ISOC 120 and software in the host 10. The software uses descriptors to transfer control information to the ISOC 120. The control information may be in the form of a frame descriptor, a pointer descriptor, or a branch descriptor depending on desired function. Descriptor logic in the software and in the ISOC 120 generate and modify the descriptors according to control measures to be taken. Such measures will be described shortly. A frame descriptor comprises a description of the packet (e.g., data length, header length, etc.). A pointer descriptor comprises a description of a data location. A branch descriptor comprises description of the descriptor location (e.g., link lists of descriptors). Information in the descriptors is used for control by the software in the host 10 of the data movement operations performed by the TX and RX LCP engines 310 and 360. The information used to process a frame to generate a TX packet header is located in the header of the frame. Referring to Figure 9A, descriptors may be provided in a single table 700 with the LCP context 140 pointing to the head of the table 700. Referring to Figure 9B, descriptors may also be arranged in a structure of linked descriptor tables 720-740. Following LCP channel initialization, the LCP context 140 points to the head of the first descriptor table 720 in the structure. Branch descriptors 750-770 are used to generate a linked list of tables 720-740 where a branch descriptor 750-770 at the end of a descriptor table 720-740 points to the beginning of another table 720-740. Referring back to Figure 9A, branch descriptors can also be used to generate a cyclic buffer where a branch descriptor 710 at the end of a table 700 points to the beginning of the same table 700. A cyclic buffer may also be used in the receive path. In this case, the LCP 140 context is initiated to point to the head of the buffer. The buffer is wrapped around when the ISOC 120 reaches its end. The software in the host 10 can

write the descriptors into the memory 60 in the host 10 (for both the receive and the transmit paths) or into the memory 250 of the adaptor 80 (for the transmit path only). Writing descriptors to the memory subsystem 210 of the adaptor 80 involves an I/O operation by the software in the host 10 and occupies the memory subsystem 210 of the adaptor 80. Writing descriptors in the memory 60 of the host 10 requires the adaptor 80 to access the memory 60 of the host 10 whenever it has to read a new descriptor. The location of the software descriptors is defined by the LCP manager 130 for each LCP channel independently. The location of the descriptors is defined according to system performance optimization. The descriptors provide flexibility in the construction of queues.

10 The RX and TX LCP engines 310 and 360 use addresses to access the descriptors in the descriptor tables and to access data buffers. An address can be either a physical address or a virtual address. The term physical address describes an address that the ISOC 120 can drive, i.e., to the bus 70. The term virtual address describes an address which is not a physical one and is used by the software or microcode. The virtual address has to pass through a mapping in order to generate the physical address. An address used by the TX and RX LCP engines 310 and 360 can have different sources as follows: pointer in the LCP channel context 160; pointer in descriptors prepared by software running on the host 10; pointer in descriptors prepared by the RX processor 160; and, pointer in descriptors prepared by the TX processor 150 (used for returning a completion message). A pointer can point to a descriptor or to a data buffer. Every address used by the TX and RX LCP engines 310 and 360 can be optionally mapped to a new address used as the physical address on the bus 70. The address mapping is done by the TX and RX LCP engines 310 and 360. The ISOC 120 uses local memory 210 to hold the translation tables. The LCP manager 130 writes the translation tables to the adaptor 80 during memory registration. The address mapping allows virtual addressing to be used for buffers or descriptor tables. The virtual addressing enables the management of virtual buffers that are physically located in more than one physical page. The address mapping also allows the host 10 to work directly with applications using virtual addresses without requiring a translation processor for the software.

30 Referring to Figure 10, shown therein is an image 800 of a buffer 880 as it appears to the software in the host 10. Also shown is a physical mapping 810 of the address as it is used to

access the memory 60 in the host 10. A virtual pointer points 820 to a location in the buffer. The buffer, in this example is a virtual buffer occupying a few noncontiguous pages 840-870 in the memory 60 of the host 10. The LCP engines 310 and 360 perform the mapping by translating the address via a translation table 830. The translation table holds a physical address pointer to the head of each physical buffer 840-870 mapped from the virtual buffer 880. Address mapping in the adapter 80 allows flexibility when mapping descriptors and data buffers in the memory 60 in the host 10. Address mapping in the adapter 80 also allows a direct connection to software buffers that use virtual addresses without requiring the software in the host 10 to perform address translation to a physical address.

- Each packet which the adapter 80 writes to the memory 60 in the host has a status associated therewith. The status allows synchronization between the adapter 80 and the software in the host 10. The status can be used to indicate different reliability levels of packets. The ISOC 120 provides the following status write backs: Transact DMA Completion indicates that a data in a TX packet has been read into the adapter 80; Reliable Transmission is returned to indicate the completion of data transmission in the network 30; Receive DMA Completion indicates completion of a receive data transfer into the memory 60; and, Reliable Reception indicates reception of a transmit packet by a destination node in the network 30.
- A TX frame descriptor includes a 2 byte status field. Status write back means that a transaction status is written back into a descriptor. The status includes a completion bit which can be polled by the software in the host 10. When the software in the host 10 finds a set completion bit, it may reuse the buffers associated with the frame defined by the frame descriptor.
- A completion queue is implemented by an RX LCP channel. The LCP channel used by the completion queue has all the flexibility and properties that can be implemented by any RX LCP channel. The TX and RX processor 150 and 160 generates status write backs to indicate reliable transmission, reliable reception, receive DMA completion, or transmit DMA completion. Different indications relating to the frame are used in different cases. For example, in the case of a reliable transmission, the TX processor 150, reads internal registers indicating the status of a packet transmission. In the case of reliable reception, the RX

WO 02/061592

-24-

PCT/IB03/06122

- processor 160 gets a completion indication as a received packet which includes an acknowledgment. In the case of a receive DMA completion, the RX processor 160 uses frame completion information. In the case of a transmit DMA completion, the TX processor 160 indicates the reception of a frame for transmission in the adapter 80. A completion queue can be used by a single TX or RX LCP channel or may shared by multiple channels. Micro code in the adapter 80 updates a status queue by initiating a frame descriptor into a command queue of the RX LCP engine 360. Referring to Figure 11, the status is transferred to the memory 60 of the host 10 via a completion status LCP 900 comprising a completion queue 920. The completion queue 900 is continuous (either physically or virtually) and is located in the memory 60 of the host 10. For example, the completion queue can be held in a continuous buffer. Entries 930 in the completion queue preferably have a fixed size. Each entry holds a pointer 940 to the head of a buffer 950 associated with a receive LCP 910. The buffer 950 is filled by the packet 960 associated with the completion status.
- 15 A TX software/driver handshake comprises an TX LCP port and an completion RX LCP port. Each LCP transmit channel uses the following data structures:
- A Doorbell entry, implemented as a memory mapped address, informs the adapter 80 of incremental requests to process descriptors and data. Each process has a unique access into a single page of memory mapped address used for Doorbell access.
 - An LCP context entry in the adapter memory space 210, containing LCP attributes and status fields.
 - 25 A structure of transmit descriptors. This structure may span across multiple physical pages in the memory 60 of the host 10. If virtual addressing is used for the descriptors, a translation table is used to move one page to the next. If physical addressing is used for the descriptors, branch descriptors are used to move from one page to the next. Transmit descriptors contain a status field that can be updated following transfer of all descriptor related data to the adapter 80.
- 30

Transmit data buffers pinned in the memory 60 of the host 10 pointed to by the pointer descriptors. If virtual addressing is used for the data buffers, a translation table converts the pointer into physical addresses used by the adapter 80 to access the memory 60 in the host 10.

5

A translation table and protection blocks in the adapter memory space 210 are used for address mapping.

Referring to Figure 12, a transmit packet flow comprises, at step 1000, software 1020 in the host 10 filling buffer 1030 with data to be transmitted. At step 1010, the software 1020 updates the descriptors 1040. The descriptors 1040 may be either in the memory 60 of the host 10 or in the memory subsystem 210 of the adapter 80. At step 1050, the software 1020 rings the Doorbell to notify the adapter 80 that new data is ready to be transmitted. At step 1060, the adapter 80 manages arbitration between requests from the different LCP channels. When a channel wins the arbitration, the adapter 80 reads the new descriptors 1040. At step 1070, the adapter 80 reads the data. At step 1080, the data is transmitted to the network 30. At step 1090, the status is updated in the descriptors 1040 or in the completion queue.

The TX LCP channel may use address translation when accessing data buffers. In this case, the data buffer is composed of multiple memory pages. As far as the present is concerned, these memory pages are in consecutive virtual memory space. However, as far as the adapter 80 is concerned, these memory pages may be in nonconsecutive physical memory space. A completion status structure contains information indicative of the status of transmitted frames. This is implemented as a separate LCP channel. The frame descriptor, which is the first descriptor for every frame, has an optional status field which can be updated after the frame has been transferred to the adapter 80.

Referring now to Figure 13, in an example of a transmit LCP channel flow, descriptors 1100 are located in the memory 60 of the host 10. Access to the descriptors 1110 and buffers 1110 storing packets 1120 requires address translation through a translation table 1130 located in the adapter 80. The buffers 1110 use contiguous space in the virtual address space of the software in the host 10. Each frame 1120 is described by two types of descriptors: a frame

20

WO 02/061592

- 26 -

PCT/JP00/06122

descriptor 1140 giving information relating the packet; and, a pointer descriptor 1150 pointing to the buffer 1110 holding the data 1120. Each packet comprises a data payload 1170 preceded by a header 1160 in the same buffer 1180.

- 5 A write transaction 1190 to the Doorbell updates the number of words 1200 available for use by the adapter 80. This information is stored in the LCP context 140. The transmit LCP context 140 includes a pointer 1210 to the head of the buffer 1110 holding the data to be transmitted. When the LCP channel wins the internal channel arbitration of the ISOC 120, the ISOC 120 reads the descriptors of the LCP channel according to the pointer 1210 in the LCP context 140. Virtual addresses, for both descriptors 1100 and buffers 1110 of the LCP channel, are translated into physical addresses using the translation table 1130 located in the memory subsystem 210 of the adapter 80. The translation table 1130 is updated by the LCP manager 140 during registration of the memory buffers. The ISOC 120 reads the data and frame headers from the buffers 1110 into the adapter 80. The frame headers 1160 are then
 10 replaced on the ISOC 1320 by a header for the network 30. The packet header and the corresponding data are then transmitted to the network 30.

- The RX LCP port is used to transfer incoming data from the ISOC 120 to the memory 60 used by a software application running on the host 10. TX LCP channels are completely controlled through descriptors initiated by the software on the host 10. RX LCP channels use descriptors
 20 from both the software on the host 10 and the ISOC 120. The descriptors initiated by the ISOC 120 are used to control the LCP channel operation to define the destination of a received frame in the memory 60 of the host 10. The descriptors initiated by the software in the host 10 can be used to define the location of buffers where the buffers were not defined through mapping in a translation table. To implement a handshake between the software in the host 10 and the adapter 80, two LCP channels are preferably used: an RX LCP channel for
 25 handling the received incoming data structure; and, an RX LCP channel for handling the completion status queue. The completion status is used by the adapter 80 to signal to the software in the host 10 that a frame transfer into the memory 60 of the host 10 is completed.
- 30 Entries are inserted into the completion queue structure in sequential addresses. Each completion status entry contains a field that is marked by the adapter 80 and probed by the software in the host 10 to check that the entry ownership has been transferred from the adapter

WO 02/061592

- 27 -

PCT/JP00/00122

80 to the software in the host 10. One or more RX LCP channels can use the same completion status queue. The sharing of the completion status queue by multiple RX LCP channels is performed by the ISOC 120.

- 5 A2: RX LCP channel requires information to indicate the destination address for an incoming packet. The ISOC 120 has two addressing for finding the location of free buffers:

Direct addressing mode refers to LCP channels that do not use pointer descriptors to point out a buffer. The destination address is defined either by microcode in the ISOC 120 or read from the context 140.

10

Indirect addressing mode refers to LCP channels that maintain pointers to data buffers in descriptor structures. The descriptors are preferably located in the memory 60 of the host 10.

15

Direct addressing substantially cuts down the latency of processing an incoming packet through the adapter 80. However, it requires registration of memory buffer by the LCP manager 130, including storage of virtual to physical translation information on the adapter 80. The software in the host 10 writes to the channels Doorbell to indicate the amount of words added to the free buffer that can be used by the channel. In direct mode, the following steps are used to determine the address of the destination buffer:

20

- A. Address A is driven as a command to the LCP engine.
- B. (Optional) Address A is mapped to address A'.
- 25 C. Address A' (if step B is executed) or A (if step B is not executed) is the base address for the destination buffer.

In indirect mode, the adapter 80 uses descriptors to find the address of the data buffers. The descriptors are managed by the software in the host 10. The descriptors are preferably located in the memory 60 of the host 10. The term indirect is used to emphasize that the adapter 80 reads additional information to define the destination address. The adapter 80 accesses this information during run-time. Indirect addressing cuts down the amount of the memory in the

30

WO 02/061592

- 28 -

PCT/IB95/00122

adapter 80 required to store translation tables. The descriptors are typically located in the memory 60 of the host 10. In indirect mode, the following steps are used to determine the address of the destination buffer:

- 5 A. Address A is given as a command to the LCP engine.
- B. (Optional) Address A is mapped to address A'.
- C. Address A' (if step B is executed) or A (if step B is not executed) is the address of the pointer descriptor.
- D. The pointer to the buffer, address B, is read from the descriptor.
- 10 E. (Optional) Address B is mapped to address B'.
- F. Address B' (if step E is executed) or B (if step E is not executed) is the basic address for the destination buffer.

Each RX LCP channel uses the following data structures:

- 15 Access to the Doorbell, implemented as a memory mapped address, informs the adapter 80 of additional data or descriptors available for the adapter 80 to write packet data.
- An LCP context entry in the memory space 210 of the adapter 80 contains LCP attributes, state, configuration, and status fields.
- 20 Descriptors pointing to memory buffers for use in indirect mode.
- A buffer in contiguous virtual address space in the memory 60 of the host 10.
- A translation table and protection blocks in the memory space 210 of the adapter 80 for address mapping.

25

The flow of receiving a packet depends on the following characteristics:

- Direct or indirect addressing mode.
- For indirect mode, descriptors are located in the memory 60 of the host 10.
- 30 For direct mode, address mapping may or may not be used during access to descriptors.
- Address mapping may or may not be used during access to buffers.

For indirect mode, address protection may or may not be used during access to descriptors.

Address protection may or may not be used during access to buffers.

- 5 These characteristics are set for each LCP channel as part of the channel's context 140 during the LCP channel initialization.

Referring to Figure 14, a flow of receive packets comprises, at step 1300, preparation by software 1310 in the host 10 of free buffers 1320 for the received data. At step 1330, in indirect mode, the software 1310 in the host 10 updates the descriptors 1340. The descriptors 1340 are located in the memory 60 of the host 10. At step 1350, the software in the host 10 rings the Doorbell to notify the adapter 80 of the free buffer space. For indirect mode, the Doorbell provides information indicative of the new descriptors 1340. For direct mode, the adapter 80 provides information indicative of added free buffer space. At this stage, the adapter 80 is ready to transfer receive data from the network 30 to the memory 60 of the host 10. Steps 1300, 1330, and 1350 are repeated whenever the software 1310 in the host 10 adds free buffers 1320 to the RX LCP channel. The 150C 120 repeats the following steps for each received packet. At step 1360, the adapter 80 receives the data. At step 1370, in indirect mode, the adapter 80 reads descriptors 1340 pointing to the location of the free data buffers 1320. At step 1380, data and headers are written into the data buffers 1340. At step 1390, status is updated in the completion queue.

Referring to Figure 15, in an example of a receive LCP channel flow, pointer descriptors are not used. Furthermore, no translation tables are used. Data buffers 1400 use contiguous space in the physical address space of software in the host 10 using the buffers 1400. Both header and data payload are written to the buffers 1400. A write instruction 1410 to the Doorbell updates the data space available for use by the adapter 80. The information is stored in the LCP context 140. The receive/completion LCP context 140 includes a pointer 1420 to the head of the buffer 1400 and an offset 1430 to the next/current address used to write new data/completion entries. When the adapter 980 receives a packet, it increments the offset 1430 to the next packet location and updates the available data space. A completion entry 1440 is added to a completion LCP 1450 upon completion of a frame reception, upon frame time-out,

WO 02/061592

- 30 -

PCT/JP00/08122

or for any other frame event that requires awareness from the LCP client 100. The completion entry 1440 contains all the information needed by the LCP client 100 to locate the frame within the LCP data buffer 1400. The software in the host 10 uses a field within the completion entry 1440 to recognize that it has been granted ownership of the completion entry 1440.

The ISOC 120 allows LCP channels to be used for moving data between the memory subsystem 210 of the adapter 80 and the memory 60 of the host 10. To transfer data from the memory 60 of the host 10 to the adapter 80 a transmit channel is used. To transfer data from the adapter 80 to the memory 60 of the host 10 a receive channel is used. When data is to be transferred from the memory 60 of the host 10 to the adapter 80 a frame descriptor includes a destination address on the bus 340 of the ISOC 120. This address defines the destination of the frame data payload. The packet header is transferred in the usual manner. This allows loading of tables and code into the memory space of the ISOC 120. To transfer data from the memory space of the ISOC 120 to the memory 60 of the host 10 using a receive channel a descriptor is initiated by the RX processor 160. The descriptor include information indicative of both destination address in the memory 60 of the host 10 and source address.

In preferred embodiments of the present invention hereinbefore described, the adapter 80 is connected to the CPU 50 and memory 60 of the host computer system 10 via the bus architecture 70. However, in other embodiments of the present invention, the adapter 80 may be integrated into the host computer system 10 independently of the bus architecture 70. For example, in other embodiments of the present invention, the adapter 80 may be integrated into the host computer system via a memory controller connected to the host memory 60.

Additionally, in preferred embodiments of the present invention hereinbefore described, the adapter 80 was implemented in the form of a pluggable adapter card for insertion into the host computer system 10. It will however be appreciated that different implementations of the adapter 80 are possible in other embodiments of the present invention. For example, the adapter 80 may be located on a mother board of the host computer system, along with the CPU 50 and the memory 60.

CLAIMS

1. Apparatus for controlling flow of data between first and second data processing systems a memory, the apparatus comprising: descriptor logic for generating a plurality of
5 descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and a descriptor table for storing the descriptors generated by the descriptor logic for access by the first and second data processing systems.
- 10 2. Apparatus as claimed in claim 1, wherein the descriptor table is stored in the first data processing system.
3. Apparatus as claimed in claim 1, wherein the descriptor table is stored in the second data processing system.
- 15 4. Apparatus as claimed in any preceding claim, wherein the descriptor logic generates a branch descriptor comprising a link to another descriptor in the descriptor table.
5. Apparatus as claimed in claim 4, wherein the descriptor table comprises a plurality of
20 descriptor lists sequentially linked together via branch descriptors therein.
6. Apparatus as claimed in claim 4, wherein the descriptor table comprises a cyclic descriptor list.
- 25 7. Apparatus as claimed in any preceding claim, wherein the first data processing system comprises a host computer system.
8. Apparatus as claimed in any preceding claim, wherein the second data processing system comprises a data communications interface for communicating data between the host
30 computer system and a data communications network.

WO 02/061592

- 32 -

PCT/JP00/00122

9. A data processing system comprising a host processing system having a memory, a data communications interface for communicating data between the host computer system and a data communications network, and apparatus as claimed in any preceding claim for controlling flow of data between the memory of the host computer system and the data communications interface
10. A method for controlling flow of data between first and second data processing systems via a memory, the method comprising: by descriptor logic, generating a plurality of descriptors including a frame descriptor defining a data packet to be communicated between a location in the memory and the second data processing system, and a pointer descriptor identifying the location in the memory; and storing the descriptors generated by the descriptor logic in a descriptor table for access by the first and second data processing systems.
11. A method as claimed in claim 10, comprising storing the descriptor table in the first data processing system.
12. A method as claimed in claim 10, comprising storing the descriptor table in the second data processing system.
13. A method as claimed in any of claims 10 to 12, comprising, by the descriptor logic, generating a branch descriptor comprising a link to another descriptor in the descriptor table.
14. A method as claimed in claim 13, comprising linking a plurality of descriptors lists together in series via branch descriptors to form the descriptor table.
15. A method as claimed in any of claims 10 to 14, wherein the first data processing system comprises a host computer system.
16. A method as claimed in any of claims 10 to 15, wherein the second data processing system comprises a data communications interface for communicating data between the host computer system and a data communications network.

WO 02/61592

PCT/JP00/0122

1/12

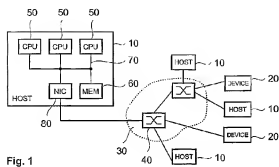


Fig. 1

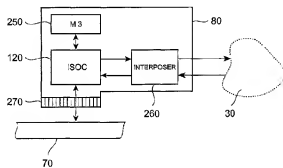
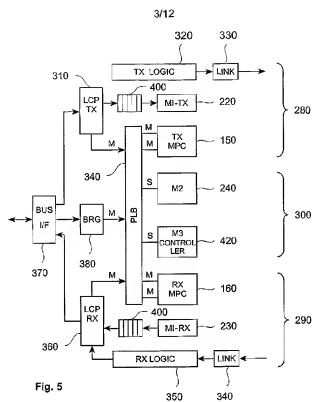


Fig. 2

SUBSTITUTE SHEET (RULE 26)

2/12





WO 02/061592

PCT/00/0122

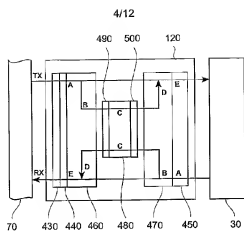


Fig. 6

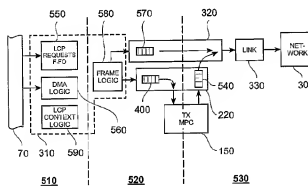


Fig. 7

SUBSTITUTE SHEET (RULE 26)

530



WO 02/061592

PCT/BR97/0122

6/12

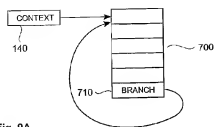


Fig. 9A

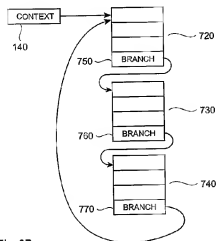


Fig. 9B

SUBSTITUTE SHEET (RULE 26)

7/12

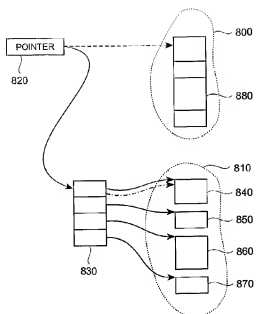


Fig. 10

WO 02/061592

PCT/00/0122

8/12

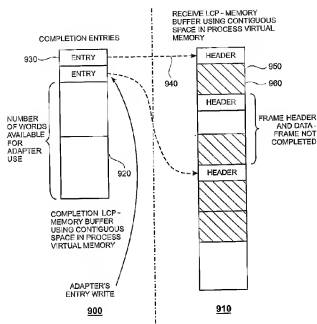


Fig. 11

SUBSTITUTE SHEET (RULE 26)

WO 02/061592

PCT/JP00/06122

9/12

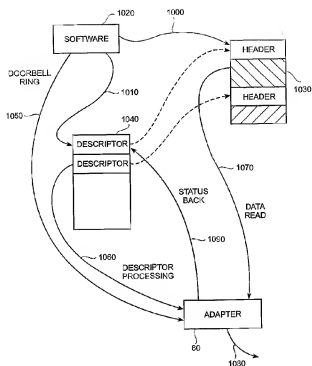


Fig. 12

SUBSTITUTE SHEET (RULE 26)

WO 02/061592

PCT/JP00/06122

10/12

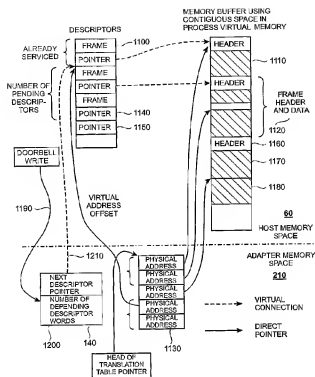


Fig. 13

SUBSTITUTE SHEET (RULE 26)

WO 02/01592

PCT/00/0122

11/12

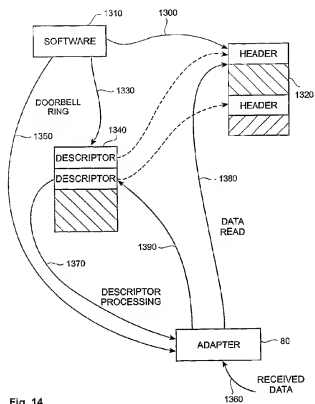


Fig. 14

SUBSTITUTE SHEET (RULE 26)

12/12

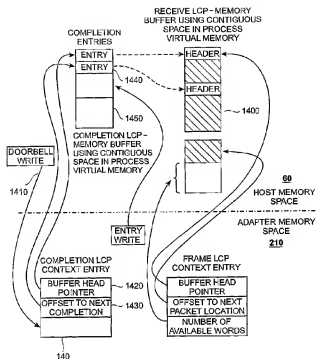


Fig. 15

SUBSTITUTE SHEET (RULE 26)

INTERNATIONAL SEARCH REPORT

Int. application No.
P. 18 01/01122

Publication number and date	Publication date	Priority date(s)	Priority date
US 5359730	A	25-10-1994	NONE
US 6528843	A	22-02-2000	GB 2324678 A, B GB 2241751 A, B JP 3065803 B2 JP 10276221 A
EP 0470876	A	12-02-1992	FR 2665040 A1 DE 69107727 01 DE 69107727 12 EP 0470876 A1 ES 2069241 T3
US 5935632	A	03-08-1996	NONE

Form P/2004-01 (September 2004) 2/8 (18/01)

フロントページの続き

(72)発明者 ビラン、ギオラ

イスラエル ズイクラン・ヤアコヴ インバル・ストリート 13

(72)発明者 ソストハイム、タル

イスラエル キルヤト・ティヴオン 36000 アロニム・ストリート 7

Fターム(参考) 5B089 GA04 KA05 KC15 KE09

5K030 GA01 HA08 HB13 LC01